The μ-diff toolbox: user guide

Xavier Antoine, Bertrand Thierry

The μ-diff toolbox: user guide

Xavier Antoine and Bertrand Thierry

January 23, 2015

version 0.9

Contents

Copyright

Université de Lorraine
Institut Elie Cartan de Lorraine, UMR CNRS 7502
F-54506 Vandoeuvre-lès-Nancy Cedex
FRANCE

Introduction to the user guide

What is μ-diff?

The toolbox μ-diff has been developed for solving two dimensional acoustic multiple scattering problems by disks based on boundary integral equations. The toolbox is a set of Matlab functions that compute accurately and efficiently the standard integral operators with in addition pre- and post-processing facilities. When the boundary integral formulation has been written mathematically, the coding part can be easily done in μ-diff. Solving the linear system is realized within the Matlab framework. No particular computational skill is needed, except basic notions in Matlab. The boundary integral operators are spectrally discretized in Fourier basis, thanks to the circular shape of the obstacles. This leads to an accurate solution and a smaller size linear systems to solve, compared to usual Boundary Element Methods (BEMs). For high-frequency and/or a large number of scatterers, an adapted algorithm is moreover used that takes advantage of the the particular structure of the matrix (block Toeplitz).

What is not μ-diff?

Even if μ-diff is designed to solve multiple scattering problem, it is not a black-box that only solves a particular problem, i.e. μ-diff does not *a priori* fix the integral equation formulation to be used for solving a particular problem. This theoretical aspect is rather let to the user. However, for users who are not fluent with boundary integral equations, this user guide provides a rapid survey on boundary integral equations (see chapter 1). Furthermore, some examples of efficient solutions based on integral equations in classical cases (Dirichlet/Neumann boundary condition, penetrable scatterers) are detailled. We expect that these elements will help a non-specialist of boundary integral equation to use μ-diff for solving multiple scattering problems.

How to use μ-diff?

It is difficult to describe the way to use μ-diff without speaking about integral equations. To simplify the presentation, we provide a basic example. Let us assume that a (circular) sound-soft scatterer Ω^- is placed within a homogeneous medium which is illuminated by an incident plane wave. The mathematical problem is to compute the scattered field solution to the Helmholtz equation in the propagation domain

$$\begin{cases} (\Delta + k^2)u = 0 & \text{in } \mathbb{R}^2 \setminus \overline{\Omega^-} \\ u = -u^{\text{inc}} & \text{on } \partial\Omega^- \\ + \text{ Sommerfeld's radiation condition at infinity} \end{cases}$$

A possible integral representation of the scattered field u is the following

$$u(\mathbf{x}) = \int_{\partial\Omega^-} G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\mathbf{y}, \qquad \forall \mathbf{x} \in \mathbb{R} \setminus \overline{\Omega^-},$$

where G is the Green function and ρ is the density, solution of a boundary integral equation, which can be e.g. the EFIE (see chapter 1)

$$L\rho = -u^{\text{inc}}, \qquad \text{with } L\rho(\mathbf{x}) = \int_{\partial\Omega^-} G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\mathbf{y}, \quad \forall \mathbf{x} \in \partial\Omega^-.$$

The problem is hence reduced to solving this boundary integral equation: knowing ρ leads to the possible computation of u at any point $\mathbf{x}$ of the propagation domain. Note that, in addition, the far-field can be obtained easily thanks to its integral representation. The theoretical part of the problem ends here and the computational part is handled now by μ-diff through the four following steps

1. Pre-processing
   (a) Creating the obstacles
   (b) Building the right-hand side ($-u^{\text{inc}}$)
2. Assembling the matrix of the integral operator(s)
3. Solving the resulting linear system
4. Post-processing: computing the far-field, near-field (at a point of the domain or on a grid), or any other physical quantity of interest ...

Most of the steps call some μ-diff functions except for the linear system solution which uses the already existing and optimized Matlab's functions for the direct (backslash operator "\") and iterative solvers (GMRES, ...). This is summarized as a diagram in Figure 1.

To whom μ-diff is designed to?

The toolbox μ-diff is designed for any scientist who needs an easy-to-use and efficient way to either solve accurately the acoustic multiple scattering problem or to compute boundary integral operators for a collection of disks. Indeed, the μ-diff toolbox can be used

- as a solver of the multiple scattering problem,
- as a framework for more theoretical studies on boundary integral equations.

What does this user guide contain?

The first chapter recalls some well-known properties of the boundary integral equations. Even if this part is mostly theoretical, some practical aspects of boundary integral equations are also provided for the impenetrable case and some examples of robust integral equations are given for the penetrable case. Moreover, some examples of usage are provided with the μ-diff toolbox. Chapter 2 explains the theory on which the μ-diff toolbox is based. More precisely, the boundary integral operators are projected in the Fourier bases. Their associated matrices are derived, in addition to other physical quantities of interest such as the far- or the near-fields. The μ-diff toolbox is detailed in chapter 3: each function is explained and examples are provided. Simple example are provided in chapter 4. Let us note that the list of available functions is given in the appendix A, in the alphabetic order, and arranged by folder location in appendix B.

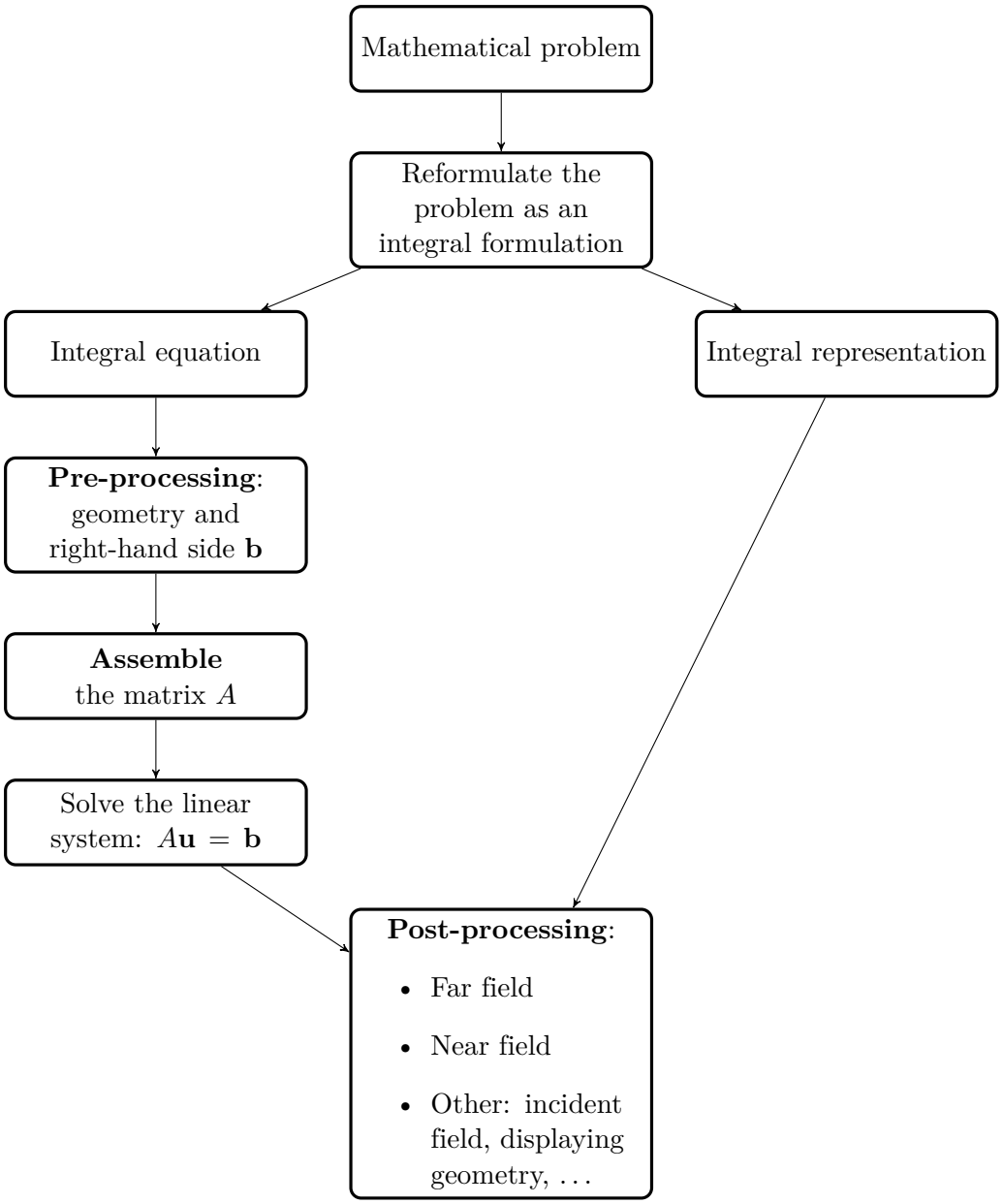


Figure 1: Schematic structure of a μ-diff script.

How to cite μ-diff?

Please cite the following reference if you use μ-diff

B. Thierry, X. Antoine, C. Chniti, H. Alzubaidi, *μ-diff: an open Matlab toolbox for computing multiple scattering problems by disks*, Computer Physics Communications, to appear, 2015.

How to install μ-diff

Requirement

The toolbox μ-diff requires the installation of the Matlab software (`http://www.mathworks.com/`), version 2011 or higher. Furthermore, μ-diff should work fine with previous versions of Matlab, however without any guarantee.

Installation

The install process is realized as follows

1. Download the μ-diff toolbox, either by using `git` with

   ```
   git clone http://mu-diff.math.cnrs.fr/git mu-diff
   ```

   or by downloading the following zip file and unzipping it in your Matlab working directory (or in any other directory that you fix yourself)

   `http://mu-diff.math.cnrs.fr/mu-diff/Download_files/mudiff.zip`

   Note that `git` should be prefered to stay up to date easily.

2. Add the μ-diff toolbox to the Matlab's path (including subfolders!), by using either the graphical interface of Matlab or the `addpath` and `save path` functions.

3. Test the μ-diff install by typing in the Matlab command window

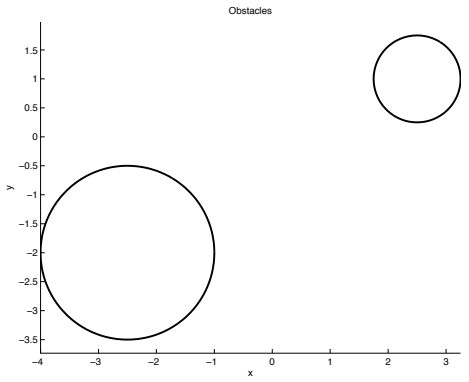
   ```
   BenchmarkDirichlet;
   ```

   This should solve the multiple scattering problem based on classical boundary integral equations, as described in chapter 1, and displays the Radar Cross Section and the history of GMRES for the various boundary integral equations, as Figure **??**.

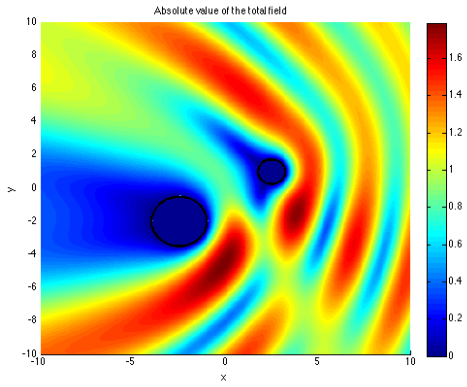
4. Other tests can be launched to check that all evaluations of the integral operators works correctly

   ```
   BenchmarkNeumann;
   BenchmarkPenetrable;
   BenchmarkCalderon;
   ```

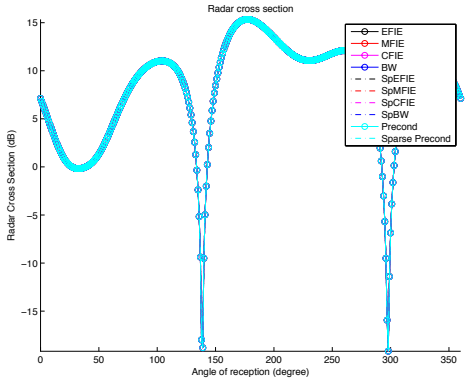
5. If everything went right: congratulation! Your μ-diff installation is working!



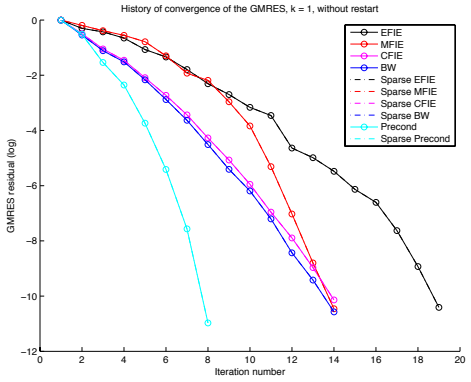
(a) Obstacles



(b) Absolute value of the total field



(c) Radar cross section



(d) History of convergence of the GMRES

Figure 2: 4 of the different figures that pop up after launching `BenchmarkDirichlet;`. The first figure shows the two obstacles and the second one the total field. The two others figures, (c) and (d) represent respectively the radar cross section for the different boundary integral equations and the history of convergence of the GMRES solver.

If you want to see some other examples by using μ-diff, you can try and launch the following time reversal experiments

- `DORT_NotPenetrable;`
- `DORT_dielectric;`

Chapter 1

Boundary integral equations: a short survey

Contents

The μ-diff toolbox is based on integral equations and uses the four standard boundary integral operators. The derivation and the choice of the integral formulation is let to the user, even if some of them are given below. In this chapter, we provide all the necessary mathematical background to solve time-harmonic wave scattering problems by (penetrable or impenetrable) circular cylinders based on integral equations. The only mathematical effort requires is that the user must of course choose the integral formulation that he wants to solve: when the integral formulation is written, then it can be solved by using μ-diff.

This chapter begins by presenting the potential theory and the four classical boundary integral operators with their main properties. The case of the scattering by disks is then studied and the boundary integral operators are projected in the Fourier bases, leading to infinite matrices but with some analytic expressions of their coefficients. We then discuss the finite dimensional approximation. The chapter concludes with the expressions of both the near- and far-fields, and the projections of the right-hand side onto the Fourier bases (incident wave).

1.1 Standard integral equation formulations in acoustic scattering

We present a way to derive standard direct integral equations in the case of non penetrable obstacles. Even if μ-diff can be used to solve the penetrable case, studying the impenetrable case is a suitable way to introduce the boundary integral operators and their properties. This section is strongly inspired by [5, 7, 15, 23].

1.1.1 Scattering problems

Let us consider a homogeneous, isotropic and non dissipative medium filling the whole space $\mathbb{R}^2$ and containing an open and bounded set Ω^-, possibly not connected but such that each component is itself simply connected. Let Γ be its boundary and $\mathbf{n}$ the unit normal vector outwardly directed to Ω^-. The domain of propagation is denoted by $\Omega^+ = \mathbb{R}^2 \setminus \overline{\Omega^-}$. When illuminated by an incident time-harmonic wave u^{inc}, the obstacle Ω^- generates a scattered wave field u that is solution to the boundary-value problem

$$\begin{cases} \Delta u + k^2 u = 0 & \text{in } \Omega^+ \\ u = -u^{\text{inc}} & \text{on } \Gamma \\ + \ u \text{ is outgoing to } \Omega^- \end{cases} \tag{1.1}$$

(The time-dependent form of the wave is assumed to be $e^{-i\omega t}$, the wavenumber $k := 2\pi/\omega$ is supposed to be real and positive.) The first equation of system (1.1) is the well-known Helmholtz equation. For the sake of conciseness, we consider a Dirichlet boundary condition on Γ. The Neumann case will be studied later. The condition "u is outgoing to Ω^-" means that u satisfies the Sommerfeld's radiation condition at infinity

$$\lim_{\|\mathbf{x}\|\to+\infty} \|\mathbf{x}\|^{1/2} \left(\nabla u \cdot \frac{\mathbf{x}}{\|\mathbf{x}\|} - iku \right) = 0,$$

where $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}$ is the euclidian norm of $\mathbb{R}^2$. Since u^{inc} is a solution of the Helmholtz equation in $\mathbb{R}^2$, then the total field $u_T = u + u^{\text{inc}}$ is solution of the following problem

$$\begin{cases} \Delta u_T + k^2 u_T = 0 & \text{in } \Omega^+ \\ u_T = 0 & \text{on } \Gamma \\ +(u_T - u^{\text{inc}}) \text{ is outgoing to } \Omega^- \end{cases} \tag{1.2}$$

It is known that, in a suitable mathematical framework, the Dirichlet scattering boundary-value problems is well-posed [14]. More precisely, we have

Theorem 1.1. *The problems (1.1) and (1.2) are uniquely solvable.*

1.1.2 Volume and boundary integral operators

Let the volume single-layer integral operator $\mathscr{L}$ be defined by (see *e.g.* [20, Theorem 6.12])

$$\begin{array}{rccll} \mathscr{L}: & H^{-1/2}(\Gamma) & \longrightarrow & H^1_{\text{loc}}(\mathbb{R}^2) & \\ & \rho & \longmapsto & \mathscr{L}\rho, \qquad \forall \mathbf{x} \in \mathbb{R}^2, & \mathscr{L}\rho(\mathbf{x}) = \displaystyle\int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{y}), \end{array}$$

and the volume double-layer integral operator $\mathscr{M}$ by

$$\begin{array}{rccll} \mathscr{M}: & H^{1/2}(\Gamma) & \longrightarrow & H^1_{\rm loc}(\mathbb{R}^2\setminus\Gamma) & \\ & \lambda & \longmapsto & \mathscr{M}\lambda, & \forall \mathbf{x}\in\mathbb{R}^2\setminus\Gamma, \mathscr{M}\lambda(\mathbf{x}) = -\displaystyle\int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\,{\rm d}\Gamma(\mathbf{y}), \end{array}$$

where the spaces $H^{-1/2}(\Gamma)$, $H^1(\mathbb{R}^2\setminus\Gamma)$, $H^1_{\rm loc}(\mathbb{R}^2\setminus\Gamma)$ are the usual Sobolev spaces and the two-dimensional Green function $G(\cdot\,,\cdot)$ is given by

$$\forall \mathbf{x},\mathbf{y}\in\mathbb{R}^2,\ \mathbf{x}\neq\mathbf{y}, \qquad G(\mathbf{x},\mathbf{y}) = \frac{i}{4}H_0^{(1)}(k\|\mathbf{x}-\mathbf{y}\|). \tag{1.3}$$

The function $H_0^{(1)}$ is the first-kind Hankel function of order zero. The way the integral operators on Γ must be understood is through the duality product between the Sobolev space $H^{1/2}(\Gamma)$ and its dual space $H^{-1/2}(\Gamma)$. However, when the data ($u^{\rm inc}$ and Γ) are smooth enough, then the scattered field u is also regular and the dual product can be identified to the (non-hermitian) inner product on $L^2(\Gamma)$

$$\langle f,g\rangle_{H^{-1/2},H^{1/2}} = \int_\Gamma f(\mathbf{x})g(\mathbf{x})\,{\rm d}\Gamma(\mathbf{x}).$$

This identification is considered throughout this paper.
Let us now define the trace $\gamma_0^\pm$ and the normal derivative trace $\gamma_1^\pm$ operators (see [13, Appendix A]), where the plus/minus signs specify whether the trace is taken from the inside of Ω^+/Ω^-. First, the trace operators $\gamma_0^\pm : H^1(\Omega^\pm)\to H^{1/2}(\Gamma)$ are defined so that, if $v\in C^\infty(\overline{\Omega^\pm})$, then

$$\gamma_0^\pm v(\mathbf{x}) = \lim_{\mathbf{z}\in\Omega^\pm\to\mathbf{x}} v(\mathbf{z}),$$

for almost every $\mathbf{x}\in\Gamma$. By introducing the space $H^1(\Omega^\pm;\Delta) := \{v\in H^1(\Omega^\pm); \Delta v\in L^2(\Omega^\pm)\}$ and the linear operators $\gamma_*^\pm : H^{1/2}(\Gamma)\to H^1(\Omega^\pm)$ such that $\gamma_0^\pm\gamma_*^\pm\varphi=\varphi$, for all $\varphi\in H^{1/2}(\Gamma)$, the normal traces $\gamma_1^\pm : H^1(\Omega^\pm;\Delta)\to H^{-1/2}(\Gamma)$ can be defined [13, Equation (A.28)]

$$\begin{array}{l} \forall v\in H^1(\Omega^\pm;\Delta), \forall\varphi\in H^{1/2}(\Gamma), \\ \qquad \left(\gamma_1^\pm v,\varphi\right)_{H^{-1/2}(\Gamma),H^{1/2}(\Gamma)} := \mp\left[\displaystyle\int_{\Omega^\pm}\Delta v(\mathbf{x})\overline{w(\mathbf{x})}\,{\rm d}\mathbf{x} + \int_{\Omega^\pm}\nabla v(\mathbf{x})\cdot\nabla\overline{w(\mathbf{x})}\,{\rm d}\mathbf{x}\right], \end{array} \tag{1.4}$$

where $w := \gamma_*^\pm\varphi$ (and thus satisfies $\gamma_0^\pm w=\varphi$). Since the quantities involved in a scattering problem do not belong to $H^1(\Omega^+)$ but rather to $H^1_{\rm loc}(\Omega^+)$, the exterior trace and normal derivative trace operators are naturally extended as $\gamma_0^+ : H^1_{\rm loc}(\Omega^+)\to H^{1/2}(\Gamma)$ and $\gamma_1^+ : H^1_{\rm loc}(\Omega^+;\Delta)\to H^{-1/2}(\Gamma)$ by $\gamma_0^+(v)=\gamma_0^+(vv')$ and $\gamma_1^+(v)=\gamma_1^+(vv')$, where v' is an arbitrarily compactly supported and infinitely differentiable function on $\overline{\Omega^+}$ which is equal to 1 in a neighborhood of Γ, and where $H^1_{\rm loc}(\Omega^+;\Delta) := \{v\in H^1_{\rm loc}(\Omega^+); \Delta v\in L^2_{loc}(\Omega^+)\}$. Remark that, when the function v is sufficiently smooth, then its normal derivative trace $\gamma_1^\pm v$, given by (1.4), belongs to $L^2(\Gamma)$ and can be written as $\gamma_1^\pm v(\mathbf{x}) = \lim_{\mathbf{z}\in\Omega^\pm\to\mathbf{x}}\nabla v(\mathbf{z})\cdot\mathbf{n}(\mathbf{x})$, for almost every $\mathbf{x}$ on Γ. Note also that, the single- and double-layer potentials, introduced previously, belong not only to $H^1_{\rm loc}(\Omega^+)\bigcup H^1(\Omega^-)$ but also to $H^1_{\rm loc}(\Omega^+;\Delta)\bigcup H^1(\Omega^-;\Delta)$ (see *e.g.* [13, §2.2]). Some well-known properties of the single- and double-layer potentials are summarized in the two following propositions. Their proof can be found for example in [20, Theorems 7.5 and 9.6] for proposition 1.2 and in [20, Theorem 6.12] for proposition 1.3.

Proposition 1.2. *For any densities $\rho\in H^{-1/2}(\Gamma)$ and $\lambda\in H^{1/2}(\Gamma)$, the single-layer potential $\mathscr{L}\rho$ and double-layer potential $\mathscr{M}\lambda$ are outgoing solutions to the Helmholtz equation in $\mathbb{R}^2\setminus\Gamma$. Moreover, the scattered field u, solution to (1.1), can be written as*

$$\forall\mathbf{x}\in\Omega^+, \qquad u(\mathbf{x}) = -\mathscr{L}(\partial_{\mathbf{n}}u|_\Gamma)(\mathbf{x}) - \mathscr{M}(u|_\Gamma)(\mathbf{x}).$$

We also have

Proposition 1.3. *The trace and the normal derivative trace of the operators $\mathscr{L}$ and $\mathscr{M}$ are given by the following relations*

$$\begin{aligned} \gamma_0^{\pm}\mathscr{L}\rho &= L\rho, & \gamma_0^{\pm}\mathscr{M}\lambda &= \left(\mp\frac{1}{2}I + M\right)\lambda, \\ \gamma_1^{\pm}\mathscr{L}\rho &= \left(\mp\frac{1}{2}I + N\right)\rho, & \gamma_1^{\pm}\mathscr{M}\lambda &= D\lambda, \end{aligned} \tag{1.5}$$

where I is the identity operator and, for $\mathbf{x} \in \Gamma, \rho \in H^{-1/2}(\Gamma)$ and $\lambda \in H^{1/2}(\Gamma)$, the four boundary integral operators are defined by

$$\begin{aligned} L: \quad H^{-1/2}(\Gamma) &\longrightarrow H^{1/2}(\Gamma), & L\rho(\mathbf{x}) &= \int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\Gamma(\mathbf{y}), \\ M: \quad H^{1/2}(\Gamma) &\longrightarrow H^{1/2}(\Gamma), & M\lambda(\mathbf{x}) &= -\int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\mathrm{d}\Gamma(\mathbf{y}), \\ N: \quad H^{-1/2}(\Gamma) &\longrightarrow H^{-1/2}(\Gamma), & N\rho(\mathbf{x}) &= \int_\Gamma \partial_{\mathbf{n_x}} G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\Gamma(\mathbf{y}) = -M^*\rho(\mathbf{x}), \\ D: \quad H^{1/2}(\Gamma) &\longrightarrow H^{-1/2}(\Gamma), & D\lambda(\mathbf{x}) &= -\partial_{\mathbf{n_x}}\int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\mathrm{d}\Gamma(\mathbf{y}). \end{aligned} \tag{1.6}$$

All along the user guide, the boundary integral operators are written with a roman letter (*e.g.* L) whereas the volume integral operators are denoted by a calligraphic letter (*e.g.* $\mathscr{L}$). According to [21, Theorems 3.4.1 and 3.4.2], the boundary integral operators L and D are invertible, providing k is not an irregular frequency. More precisely, we have

Theorem 1.4. *Let $F_D(\Omega^-)$ (resp. $F_N(\Omega^-)$) be the countable set of positive wavenumbers k accumulating at infinity such that the interior homogeneous Dirichlet (resp. Neumann) problem*

$$\begin{cases} -\Delta v = k^2 v & \text{in } \Omega^-, \\ v = 0 \,(\text{resp. } \partial_{\mathbf{n}} v = 0) & \text{on } \Gamma, \end{cases} \tag{1.7}$$

admits non-trivial solutions. Then, the operator L (resp. D) realizes an isomorphism from $H^{-1/2}(\Gamma)$ into $H^{1/2}(\Gamma)$ (resp. from $H^{1/2}(\Gamma)$ into $H^{-1/2}(\Gamma)$) if and only if $k \notin F_D(\Omega^-)$ (resp. $k \notin F_N(\Omega^-)$).

These irregular frequencies k of $F_D(\Omega^-)$ (resp. of $F_N(\Omega^-)$) are exactly the square-roots of the eigenvalues of the Laplacian operator $(-\Delta)$ for the homogeneous interior Dirichlet (resp. Neumann) problem. In the multiple scattering case, that is when $\Omega^- = \bigcup_{p=1}^M \Omega_p^-$ is multiply connected, the following equalities hold true

$$F_D(\Omega^-) = \bigcup_{p=1}^M F_D(\Omega_p^-) \qquad \text{and} \qquad F_N(\Omega^-) = \bigcup_{p=1}^M F_N(\Omega_p^-). \tag{1.8}$$

Throughout the paper, $F_{DN}(\Omega^-)$ denotes the set of all irregular frequencies

$$F_{DN}(\Omega^-) = F_D(\Omega^-) \bigcup F_N(\Omega^-). \tag{1.9}$$

1.1.3 Direct integral equations

Generalities

This section details the way of deriving direct integral equations, as described in [5, 7, 15, 23]. This approach is nonstandard but has advantages that appear later in the user guide in section 1.2.2. The principle is to write the total field u_T as a linear combination of a single- and double-layer potentials

$$u_T(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + \mathscr{M}\lambda(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega^+, \tag{1.10}$$

where (λ, ρ) is now the unknown of the problem. Thanks to proposition 1.2, such an expression ensures that both u_T is solution of the Helmholtz equation in Ω^+ and $(u_T - u^{\text{inc}})$ is outgoing. Following [7], an integral equation is said to be direct when the densities (λ, ρ) have a physical meaning. Indeed, for these integral equations, they are exactly the Cauchy data $(-u_T|_\Gamma, -\partial_{\mathbf{n}} u_T|_\Gamma)$. However, this is not a choice but a consequence of the construction of the integral equation. In electromagnetic scattering, direct and indirect integral equations are more often referred to as respectively *field* and *source* integral equations (see *e.g.* Harington and Mautz [16, 19] or Borel [9]).
From now on, the problem, with unknown (λ, ρ), has only one equation given by the Dirichlet boundary condition on Γ. To obtain a second equation, a fictitious interior wave u_T^-, defined in Ω^-, is introduced through

$$u_T^-(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + \mathscr{M}\lambda(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega^-. \tag{1.11}$$

Remark that, on the one hand u_T^- is a solution of the Helmholtz equation in Ω^- and, on the other hand, due to the trace relations (1.5), the couple of unknowns (λ, ρ) satisfies the well-known *jump-relations*

$$\begin{cases} \lambda = u_T^-|_\Gamma - u_T|_\Gamma, \\ \rho = \partial_{\mathbf{n}} u_T^-|_\Gamma - \partial_{\mathbf{n}} u_T|_\Gamma. \end{cases} \tag{1.12}$$

As the wave u_T^- is fictitious, it does not act on the solution u_T of the scattering problem. As a consequence, the boundary condition on Γ imposed to u_T^- has no influence on u_T. Let this constraint be represented by an operator A such that u_T^- is the solution of the following interior problem

$$\begin{cases} \Delta u_T^- + k^2 u_T^- = 0 & \text{in } \Omega^-, \\ A u_T^- = 0 & \text{on } \Gamma. \end{cases} \tag{1.13}$$

To build a direct integral equation, the operator A is chosen such that the field u_T^- vanishes in Ω^-. Supposing that such an operator exists, then the following equalities hold on the boundary Γ

$$\begin{cases} u_T^-|_\Gamma = 0, \\ \partial_{\mathbf{n}} u_T^-|_\Gamma = 0. \end{cases}$$

Consequently and thanks to the Dirichlet boundary condition $u_T|_\Gamma = 0$, the jump relations (1.12) read as

$$\begin{cases} \lambda = 0, \\ \rho = -\partial_{\mathbf{n}} u_T|_\Gamma, \end{cases}$$

Therefore, both the fictitious interior field u_T^- and the total field u_T can be composed by a single-layer potential only

$$\begin{cases} u_T(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}), & \forall \mathbf{x} \in \Omega^+, \\ u_T^-(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}), & \forall \mathbf{x} \in \Omega^-. \end{cases}$$

The unknown ρ is finally obtained through the solution of the (direct) integral equation $Au_T^- = 0$, which can be written as

$$A\mathscr{L}\rho = -Au^{\text{inc}}. \tag{1.14}$$

Both the expression and the nature of the integral equation (1.14) depend on the boundary condition imposed to u_T^-, represented here by the operator A. The next steps introduce three usual direct integral equations. The proofs are not provided and can be found for example in [7] or [23].

EFIE (*Electric Field Integral Equation*)

For this first integral equation, the operator A is the interior trace operator γ_0^- on Γ. Thanks to the continuity on Γ of the single-layer integral operator $\mathscr{L}$ (see equation (1.5)), the boundary integral equation (1.14) becomes

$$L\rho = -u^{\text{inc}}|_\Gamma. \tag{1.15}$$

Due to theorem 1.4, this first-kind integral equation, called *Electric Field Integral Equation* (EFIE), is well-posed and equivalent to the scattering problem (1.2) except for Dirichlet irregular frequencies.

Proposition 1.5. *If $k \notin F_D(\Omega^-)$, then the function $\mathscr{L}\rho + u^{inc}$ is solution to the scattering problem (1.2) if and only if ρ is the solution of the EFIE (1.15).*

When $k \in F_D(\Omega^-)$, the integral operator L is no longer bijective but is still one-to-one. It can be shown that the kernel of the operator L is a subset of the kernel of the operator $\mathscr{L}$. Consequently, for every solution $\widetilde{\rho}$ of the EFIE, the associated single-layer potential $\mathscr{L}\widetilde{\rho} + u^{\text{inc}}$ is still the solution of the scattering problem (1.2).

MFIE (*Magnetic Field Integral Equation*)

Another possibility is to choose $A = \gamma_1^-$, the interior normal derivative trace. Using the traces formulas (1.5), the integral equation (1.14) becomes

$$\left(\frac{1}{2}I + N\right)\rho = -\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma. \tag{1.16}$$

This Fredholm second-kind integral equation, named *Magnetic Field Integral Equation* (MFIE), is well-posed and equivalent to the scattering problem (1.2) as far as k is not an irregular Neumann frequency.

Proposition 1.6. *If $k \notin F_N(\Omega^-)$, then the quantity $\mathscr{L}\rho + u^{inc}$ is the solution of the scattering problem (1.2) if and only if ρ is the solution of the MFIE (1.16).*

For any irregular frequency k of $F_N(\Omega^-)$, the operator $\left(\frac{1}{2}I + N\right)$ is no longer one-to-one. In that case and unlike the EFIE, the single-layer potential $\mathscr{L}\widetilde{\rho} + u^{\text{inc}}$ based on a solution $\widetilde{\rho}$ of the MFIE is not guaranteed to be the solution of the scattering problem (1.2).

CFIE (*Combined Field Integral Equation*)

To avoid the irregular frequencies problem, Burton and Miller [11] considered a linear combination of the EFIE and the MFIE by imposing a Fourier-Robin boundary condition to u_T^- on the boundary Γ

$$A = (1-\alpha)\gamma_1^- + \alpha\eta\gamma_0^-, \tag{1.17}$$

with

$$0 < \alpha < 1 \qquad \text{and} \qquad \Im(\eta) \neq 0, \tag{1.18}$$

where $\Im(\eta)$ is the imaginary part of the complex number η. Hence, the boundary integral equation (1.14) reads as

$$\left[(1-\alpha)\left(\frac{1}{2}I + N\right) + \alpha\eta L\right]\rho = -\left[(1-\alpha)\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma + \alpha\eta u^{\text{inc}}|_\Gamma\right]. \tag{1.19}$$

This *Combined Field Integral Equation* (CFIE, denomination of Harrington and Mautz [16] in electromagnetism) or Burton-Miller integral equation [11] is well-posed for any frequency k.

Proposition 1.7. *For any $k > 0$ and for any complex-valued numbers α and η satisfying condition (1.18), the function $\mathscr{L}\rho + u^{inc}$ is the solution of the scattering problem if and only if ρ is the solution of the CFIE (1.2).*

1.1.4 Brakhage-Werner indirect integral equation

The indirect Brakhage-Werner Integral Equation (BWIE) is derived now. Let us first start by introducing some notations. The total field u_T is here sought as a linear combination of a single- and a double-layer potentials applied to the density $\psi \in H^{1/2}(\Gamma)$

$$u_T = u^{\text{inc}} + \mathscr{L}_{\text{BW}}\psi,$$

where the operator $\mathscr{L}_{\text{BW}}$ with parameter η_{BW} is given by

$$\begin{aligned}\forall \mathbf{x} \in \mathbb{R}^d \setminus \Gamma, \mathscr{L}_{\text{BW}}\psi(\mathbf{x}) = (-\eta_{\text{BW}}\mathscr{L} - \mathscr{M})\psi(\mathbf{x}) = \\ \int_\Gamma \Big(\partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y}) - \eta_{\text{BW}} G(\mathbf{x},\mathbf{y})\Big)\psi(\mathbf{y})\, \mathrm{d}\Gamma(\mathbf{y}),\end{aligned} \tag{1.20}$$

with $\Im(\eta) \neq 0$. The integral equation is obtained by applying the exterior trace γ_0^+ to u_T. Indeed, the Dirichlet boundary condition $\gamma_0^+ u_T = 0$ and the trace relations (1.5) directly give the Brakhage-Werner integral equation with ψ as unknown

$$L_{\text{BW}}\psi = -u^{\text{inc}}|_\Gamma, \tag{1.21}$$

with

$$L_{\text{BW}} = \left(-\eta L - M + \frac{1}{2}I\right).$$

This second-kind integral equation does not suffer from irregular frequency [10].

Proposition 1.8. *For all $k > 0$, the quantity $\mathscr{L}_{BW}\psi + u^{inc}$ is the solution to (1.2) if and only if ψ is the solution of the Brakhage-Werner integral equation (1.21).*

A numerical study concerning the optimal choice of the parameter η_{BW} (see relation (1.20)) is proposed in [17] in the case of a single spherical or circular obstacle of radius R. For a Dirichlet boundary condition, the choice $\eta_{\text{BW}} = i/2\max(1/R, k)$ leads to a reasonable condition number of the matrix of the linear system associated to the Brakhage-Werner integral equation, for a sufficiently high frequency. Recent works have been developed on how to choose this parameter for much more general domains, see for example [12, §6] and [13, §5.1] for the case of large k and [8, §2.6 and §2.7] for the case of small frequency k. Note also that, according to [13, Remark 2.24], these results apply to both L_{BW} and the CFIE operator, since when $\alpha = 1/2$, these operators are adjoints (up to a factor of $1/2$) in the real L^2 inner product. Other generalizations of these equations, when η_{BW} is an operator, are available for example in [1, 3, 4]. They provide a clear background concerning the generalization and improvement of both the CFIE and BWIE.

1.1.5 Neumann boundary condition

Consider now the scattering of a wave by a sound-hard obstacle (Neumann boundary condition)

$$\begin{cases} \Delta u + k^2 u = 0 & \text{in } \Omega^+, \\ \partial_{\mathbf{n}} u = -\partial_{\mathbf{n}} u^{\text{inc}} & \text{on } \Gamma, \\ + \ u \text{ is outgoing.} \end{cases}$$

The scattered field u is sought in the form of a linear combination between a single- and a double-layer potentials

$$u(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + \mathscr{M}\lambda(\mathbf{x}), \qquad \mathbf{x} \in \Omega^+.$$

To build the direct integral equations, let us introduce the interior fictitious wave $u_T^- = \mathscr{L}\rho + \mathscr{M}\lambda + u^{\text{inc}}$ defined in Ω^-. On Γ, a boundary condition is hence enforced to the field u_T^- such that it vanishes in Ω^-. In that case the jump relations (1.12) lead to

$$\begin{cases} \lambda = -u_T|_\Gamma, \\ \rho = -\partial_{\mathbf{n}} u_T|_\Gamma = 0. \end{cases}$$

The Neumann boundary condition then makes the density ρ vanishing. The interior u_T^- and exterior u_T fields are obtained through a double-layer potential

$$u_T^-(\mathbf{x}) = \mathscr{M}\lambda(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega^-,$$

$$u_T = \mathscr{M}\lambda(\mathbf{x}) + u^{\text{inc}}(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega^+.$$

As previously, the boundary condition applied to u_T^- is the integral equation to solve. Here are listed the four integral equations obtained in addition with their properties

- **EFIE**: applying a homogeneous Neumann boundary condition to the wave u_T^-

  $$\partial_{\mathbf{n}} u_T^-|_\Gamma = 0, \tag{1.22}$$

  leads to the EFIE for the Neumann boundary condition

  $$D\lambda = -\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma.$$

  This first-kind integral equation is well-posed and equivalent to the scattering problem as long as k is not an irregular frequency for the interior homogeneous Neumann problem. If $k \in F_N(\Omega^-)$ however, then after reconstruction, the solution obtained by the EFIE is correct.

- **MFIE**: applying a Dirichlet boundary condition to the field u_T^-

  $$u_T^-|_\Gamma = 0,$$

  leads to the MFIE for the Neumann boundary condition

  $$\left(\frac{1}{2} I + M\right) \lambda = -u^{\text{inc}}|_\Gamma. \tag{1.23}$$

  This second-kind Fredholm integral equation (the operator M is compact) is well-posed. It is equivalent to the scattering problem if k is not an interior resonance for the Dirichlet boundary-value problem. In that case, the solution obtained from the MFIE is not correct and must not be used for a practical computation.

- **CFIE**: applying the mixed boundary condition (1.17) to the wave u_T^-

$$(1-\alpha)\partial_{\mathbf{n}} u_T^-|_\Gamma + \alpha\eta u_T^-|_\Gamma = 0,$$

with

$$0 < \alpha < 1 \qquad \text{et} \qquad \Im(\eta) \neq 0,$$

leads to the CFIE for a Neumann boundary condition

$$\left[(1-\alpha)D + \alpha\eta\left(\frac{1}{2}I + M\right)\right]\lambda = -\left[(1-\alpha)\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma + \alpha\eta u^{\text{inc}}|_\Gamma\right]. \tag{1.24}$$

This first-kind integral equation is uniquely solvable for any wavenumber k and is equivalent to the scattering problem.

- **BWIE**: the wave u_T is represented as

$$u = (-\mathscr{L} - \eta\mathscr{M})\psi,$$

with $\Im(\eta) \neq 0$. Applying the boundary condition on Γ gives the BWIE

$$\left(-\eta D + \frac{1}{2}I - N\right)\psi = -\partial_{\mathbf{n}} u^{inc}|_\Gamma. \tag{1.25}$$

This first-kind integral equation is also uniquely solvable for every $k > 0$ and equivalent to the scattering problem.

1.1.6 Summary

The following table summarizes the main properties of the different integral equations for a Dirichlet (respectively Neumann) boundary condition

Dirichlet (resp. Neumann) boundary condition			
Integ. Eq.	Nature	Uniquely solvable for ...	Correct physical solution? ...
EFIE	1^{st} kind	$k \notin F_D(\Omega^-)$ (resp. $F_N(\Omega^-)$)	yes
MFIE	2^{nd} kind	$k \notin F_N(\Omega^-)$ (resp. $F_D(\Omega^-)$)	no
CFIE	2^{nd} (1^{st}) kind	$k > 0$	yes
BWIE	2^{nd} (1^{st}) kind	$k > 0$	yes

1.2 Multiple scattering case

1.2.1 A more explicit writing of the integral equation formulations

The domain Ω^- is now supposed to be a collection of M disjoint bounded open sets Ω_p^- of $\mathbb{R}^2$, $p = 1, \ldots, M$, such that each domain $\mathbb{R}^2 \setminus \overline{\Omega_p^-}$ is connected. In the user guide, single-scattering designates scattering in a medium containing only one scatterer whereas multiple scattering is used for a medium containing more than one obstacle. We focus on the multiple scattering case, i.e. $M \geq 2$. Since Ω^- is composed of M disjoint obstacles Ω_p^-, $p = 1, \ldots, M$, the single-layer volume integral operator $\mathscr{L}$ can be written as the sum of M operators $\mathscr{L}_q$, $q = 1, \ldots, M$, defined by

$$\begin{array}{rccl} \mathscr{L}_q : & H^{-1/2}(\Gamma_q) & \longrightarrow & H^1_{\text{loc}}(\mathbb{R}^2) \\ & \rho_q & \longmapsto & \mathscr{L}_q\rho_q, \qquad \forall \mathbf{x} \in \mathbb{R}^2, \quad \mathscr{L}_q\rho_q(\mathbf{x}) = \displaystyle\int_{\Gamma_q} G(\mathbf{x},\mathbf{y})\rho_q(\mathbf{y})\,\mathrm{d}\mathbf{y}. \end{array} \tag{1.26}$$

Therefore, the single-layer potential can be decomposed as follows

$$\forall \rho \in H^{-1/2}(\Gamma), \qquad \mathscr{L}\rho = \sum_{q=1}^{M} \mathscr{L}_q \rho_q, \qquad \text{with } \rho_q = \rho|_{\Gamma_q}. \tag{1.27}$$

By introducing the operators $L_{p,q}$, for $p, q = 1, \ldots, M$, defined on $H^{-1/2}(\Gamma_q)$ by

$$\forall \rho_q \in H^{-1/2}(\Gamma_q), \qquad L_{p,q}\rho_q = (\mathscr{L}_q \rho_q)|_{\Gamma_p},$$

that is

$$\forall \rho_q \in H^{-1/2}(\Gamma_q), \forall \mathbf{x} \in \Gamma_p, \qquad L_{p,q}\rho_q(\mathbf{x}) = \int_{\Gamma_q} G(\mathbf{x}, \mathbf{y}) \rho_q(\mathbf{y}) \, \mathrm{d}\mathbf{y}, \tag{1.28}$$

then the EFIE can be written in the following matrix form

$$\begin{bmatrix} L^{1,1} & L^{1,2} & \ldots & L^{1,M} \\ L^{2,1} & L^{2,2} & \ldots & L^{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ L^{M,1} & L^{M,2} & \ldots & L^{M,M} \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_M \end{bmatrix} = - \begin{bmatrix} u^{\mathrm{inc}}|_{\Gamma_1} \\ u^{\mathrm{inc}}|_{\Gamma_2} \\ \vdots \\ u^{\mathrm{inc}}|_{\Gamma_M} \end{bmatrix}. \tag{1.29}$$

The same procedure can be applied to the other volume operator $\mathscr{M}$ and the other boundary integral operators M, N and D to obtain

$$\begin{array}{lcccll} L^{p,q}: & H^{-1/2}(\Gamma_q) & \longrightarrow & H^{1/2}(\Gamma_p), & L\rho_q(\mathbf{x}) & = \displaystyle\int_{\Gamma_q} G(\mathbf{x}, \mathbf{y}) \rho_q(\mathbf{y}) \mathrm{d}\Gamma_q(\mathbf{y}), \\ M^{p,q}: & H^{1/2}(\Gamma_q) & \longrightarrow & H^{1/2}(\Gamma_p), & M\lambda_q(\mathbf{x}) & = -\displaystyle\int_{\Gamma_q} \partial_{\mathbf{n_y}} G(\mathbf{x}, \mathbf{y}) \lambda_q(\mathbf{y}) \mathrm{d}\Gamma_q(\mathbf{y}), \\ N^{p,q}: & H^{-1/2}(\Gamma_q) & \longrightarrow & H^{-1/2}(\Gamma_p), & N\rho_q(\mathbf{x}) & = \displaystyle\int_{\Gamma_q} \partial_{\mathbf{n_x}} G(\mathbf{x}, \mathbf{y}) \rho_q(\mathbf{y}) \mathrm{d}\Gamma_q(\mathbf{y}), \\ D^{p,q}: & H^{1/2}(\Gamma_q) & \longrightarrow & H^{-1/2}(\Gamma_p), & D\lambda_q(\mathbf{x}) & = -\partial_{\mathbf{n_x}} \displaystyle\int_{\Gamma_q} \partial_{\mathbf{n_y}} G(\mathbf{x}, \mathbf{y}) \lambda_q(\mathbf{y}) \mathrm{d}\Gamma_q(\mathbf{y}). \end{array}$$

1.2.2 Single-scattering preconditioning

Let us introduce the *single-scattering operator* of the EFIE $\widehat{L}$ as the diagonal part of the operator L defined by

$$\widehat{L} = \begin{bmatrix} L^{1,1} & 0 & \ldots & 0 \\ 0 & L^{2,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & L^{M,M} \end{bmatrix}. \tag{1.30}$$

Let k be a wavenumber k that is not an interior resonance ($k \notin F_D(\Omega^-)(\Omega)$), implying that L and $\widehat{L}$ are both invertible. The single-scattering preconditioner then simply consists in $\widehat{L}$

$$\widehat{L}^{-1} = \begin{bmatrix} (L^{1,1})^{-1} & 0 & \ldots & 0 \\ 0 & (L^{2,2})^{-1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (L^{M,M})^{-1} \end{bmatrix}.$$

The associated preconditioned EFIE becomes

$$\widehat{L}^{-1} L \rho = -\widehat{L}^{-1} u^{\mathrm{inc}}|_{\Gamma}, \tag{1.31}$$

where the operator $\widehat{L}^{-1}L$ has the following matrix form

$$\widehat{L}^{-1}L = \begin{bmatrix} I & (L^{1,1})^{-1}L^{1,2} & \dots & (L^{1,1})^{-1}L^{1,M} \\ (L^{2,2})^{-1}L^{2,1} & I & \dots & (L^{2,2})^{-1}L^{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ (L^{M,M})^{-1}L^{M,1} & (L^{M,M})^{-1}L^{M,2} & \dots & I \end{bmatrix}, \tag{1.32}$$

with I the identity operator. Note that this preconditioning accelerates the convergence rate of an iterative solver, like e.g. the GMRES. The single scattering preconditioner can be applied to the other integral equations and the following result holds [24]

Proposition 1.9. *When preconditioned by their single-scattering preconditioners, the EFIE, MFIE and CFIE become identical and similar to the preconditioned BWIE (equal up to an invertible operator). In other words, after being preconditioned, the four integral equations lead to the same convergence rate when an iterative solver is applied.*

This proposition shows in particular that there is no need in computing the single-scattering preconditioned versions of all the integral equations. Because the resulting operator exhibits a good convergence rate for multiple scattering, it is hard coded in μ-diff for both the Dirichlet and the Neumann cases. They are based on the single-layer potential for the Dirichlet case and the double-layer potential for the Neumann case (EFIE, MFIE or CFIE in both cases).

1.3 Mixing Dirichlet and Neumann boundary conditions

Let us assume that the collection of M obstacles is such that there is M_D sound-soft obstacles and M_N sound-hard scatterers, with $M_D + M_N = M$. Without loss of generality, let the sound-soft obstacles Ω_p^- be numbered for $p = 1, \dots, M_D$, and the sound-hard ones for $p = M_D + 1, \dots, M_D + M_N$. The problem then reads as

$$\begin{cases} (\Delta + k^2)u &= 0 & \text{in } \Omega^+, \\ u &= -u^{\text{inc}} & \text{on } \Gamma_p, \quad p = 1, \dots, M_D, \\ \partial_{\mathbf{n}} u &= -\partial_{\mathbf{n}} u^{\text{inc}} & \text{on } \Gamma_p, \quad p = M_D + 1, \dots, M_D + M_N, \\ + & \ u \text{ is outgoing to } \Omega^-. \end{cases}$$

The field is then represented as a linear combination of single- and double-layer potentials

$$u = \sum_{q=1}^{M} \mathscr{L}_q \rho_q + \sum_{q=1}^{M} \mathscr{M}_q \lambda_q,$$

where $\mathscr{L}_q$ is given by (1.33) and $\mathscr{M}_q$ by

$$\begin{array}{rccl} \mathscr{M}_q : & H^{1/2}(\Gamma_q) & \longrightarrow & H^1_{\text{loc}}(\mathbb{R}^2) \\ & \lambda_q & \longmapsto & \mathscr{M}_q \lambda_q, \qquad \forall \mathbf{x} \in \mathbb{R}^2, \quad \mathscr{M}_q \lambda_q(\mathbf{x}) = \displaystyle\int_{\Gamma_q} \partial_{\mathbf{n}_\mathbf{y}} G(\mathbf{x}, \mathbf{y}) \lambda_q(\mathbf{y}) \, \mathrm{d}\mathbf{y}. \end{array} \tag{1.33}$$

A possible way to solve this problem is to apply the CFIE on both collections. To derive the CFIE, a fictitious field is introduced and the mixed condition (1.17) is applied on it. Following the theory previously described, this would lead to

$$\begin{cases} \lambda_q = 0 & \text{for } q = 1, \dots, M_D \\ \rho_q = 0 & \text{for } q = M_D + 1, \dots, M_D + M_N, \end{cases}$$

and the scattered field u is represented as

$$u = \sum_{q=1}^{M_D} \mathscr{L}_q \rho_q + \sum_{q=M_D+1}^{M_D+M_N} \mathscr{M}_q \lambda_q.$$

To simplify, let us assume now that there are only two obstacles, the first one being sound-soft and the second one being sound-hard. The extension to M various scatterers is straightforward. The boundary condition (1.17) applied to the interior fictitious field leads to the integral operator

$$\begin{pmatrix} A_1\mathscr{L}_1 & A_1\mathscr{M}_2 \\ A_2\mathscr{L}_1 & A_2\mathscr{M}_2 \end{pmatrix},$$

where A_p represents the boundary condition (1.17) on Γ_p only

$$A_p = (1-\alpha)\gamma_{1,p}^- + \alpha\eta\gamma_{0,p}^-.$$

Applying the trace formulas leads to the system

$$\begin{pmatrix} (1-\alpha)\left(\frac{I}{2} + N_{1,1}\right) + \alpha\eta L_{1,1} & (1-\alpha)D_{2,1} + \alpha\eta M_{2,1} \\ (1-\alpha)N_{1,2} + \alpha\eta L_{1,2} & (1-\alpha)D_{2,2} + \alpha\eta\left(\frac{I}{2} + M_{2,2}\right) \end{pmatrix} \begin{pmatrix} \rho_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

with

$$b_p = (1-\alpha)u^{\text{inc}}|_{\Gamma_p} + \alpha\eta\partial_{\mathbf{n}}u^{\text{inc}}|_{\Gamma_p}.$$

For M obstacles, the same procedure can be applied to obtain

$$\left(\frac{I}{2} + A\right)\varphi = b, \tag{1.34}$$

where

- the matrix A of size $M \times M$ is such that

$$A^{p,q} = \begin{cases} (1-\alpha)N^{p,q} + \alpha\eta L^{p,q}, & \text{if } q \le M_D, \\ (1-\alpha)D^{p,q} + \alpha\eta M^{p,q}, & \text{if } q > M_D, \end{cases} \tag{1.35}$$

- the unknown vector $\varphi = (\varphi_q)_{1\le q\le M}$ collects the contributions of the elementary densities

$$\varphi_q = \begin{cases} \rho_q, & \text{if } 1 \le q \le M_D, \\ \lambda_q, & \text{if } M_D + 1 \le q \le N_D, \end{cases}$$

- the right-hand side $b = (b_p)_{1\le p\le M}$ is given by

$$b_p = (1-\alpha)u^{\text{inc}}|_{\Gamma_p} + \alpha\eta\partial_{\mathbf{n}}u^{\text{inc}}|_{\Gamma_p}.$$

1.4 The penetrable case

1.4.1 The boundary-value problem

Let assume that the obstacles are now penetrable with a contrast function n such that

$$n(\mathbf{x}) = \begin{cases} n_p & \text{if } \mathbf{x} \in \Omega_p^-, \\ 0 & \text{otherwise,} \end{cases}$$

where n_p are constant for each obstacle. The boundary-value problem to solve is given by: find the total field u_T such that

$$\left\{\begin{array}{rcll}(\Delta+k^2)u_T &=& 0 & \text{in } \Omega^+,\\ (\Delta+k^2n^2)u_T &=& 0 & \text{in } \Omega^-,\\ u_T &=& 0 & \text{on } \Gamma,\\ \partial_{\mathbf{n}}u_T &=& 0 & \text{on } \Gamma,\\ (u_T-u^{\text{inc}}) & \text{is outgoing to } \Omega^-. & &\end{array}\right.$$

1.4.2 An example of integral equation for the penetrable case

To rewrite this problem as a boundary integral equation, let the field u_T be rewritten as

$$u_T=\begin{cases}u^+ + u^{\text{inc}} & \text{in } \Omega^+,\\ u^- & \text{in } \Omega^-,\end{cases}$$

where u^+ and u^- are solutions of the coupled problems

$$\left\{\begin{array}{rcll}(\Delta+k^2)u^+ &=& 0 & \text{in } \Omega^+,\\ (\Delta+k^2n^2)u^- &=& 0 & \text{in } \Omega^-,\\ (u^+-u^-)|_\Gamma &=& -u^{\text{inc}}|_\Gamma & \text{on } \Gamma,\\ (\partial_{\mathbf{n}}u^+-\partial_{\mathbf{n}}u^-)|_\Gamma &=& -\partial_{\mathbf{n}}u^{\text{inc}}|_\Gamma & \text{on } \Gamma,\\ u^+ & \text{is outgoing to } \Omega^-. & &\end{array}\right.$$

Following the EFIE procedure, let the two fields u^+ and u^- be expressed as a single-layer potential only

$$\begin{cases}u^+(\mathbf{x})=\mathscr{L}^+\rho^+(\mathbf{x})=\displaystyle\int_\Gamma G^+(\mathbf{x},\mathbf{y})\rho^+(\mathbf{y})\,\mathrm{d}\mathbf{y} & \text{in } \Omega^+,\\ u^-(\mathbf{x})=\mathscr{L}^-\rho^-(\mathbf{x})=\displaystyle\int_\Gamma G^-(\mathbf{x},\mathbf{y})\rho^-(\mathbf{y})\,\mathrm{d}\mathbf{y} & \text{in } \Omega^-,\end{cases}$$

where

$$G^\pm=\frac{i}{4}H_0^{(1)}(k^\pm\|\mathbf{x}-\mathbf{y}\|)$$

corresponds to the Green function with respectively the wavenumber $k^+ := k$ and $k^- := kn$ (note that k^- can differ from one obstacle to another). Applying the transmission boundary conditions for the trace and normal derivative trace on Γ leads to the following couple system of boundary integral equations

$$\begin{pmatrix} L^+ & -L^- \\ -\frac{I}{2}+N^+ & -\frac{I}{2}-N^-\end{pmatrix}\begin{pmatrix}\rho^+\\ \rho^-\end{pmatrix}=\begin{pmatrix}-u^{\text{inc}}|_\Gamma\\ -\partial_{\mathbf{n}}u^{\text{inc}}|_\Gamma\end{pmatrix}. \tag{1.36}$$

The upper index $\pm$ appearing on the boundary integral operators specifies which Green function is used in their expression. For instance, we have

$$L^\pm\rho=\int_\Gamma G^\pm(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\,\mathrm{d}\mathbf{y}.$$

It can be proved that this integral equation is well-posed if k is not an irregular frequency of the interior homogeneous Dirichlet problem. The example `BenchmarkPenetrable.m` located in `Examples/Benchmark` solves the penetrable scattering problem by using this integral equation. Actually, the program has been written in the dielectric case (not the acoustic one), and thus,

the transmission condition on the normal derivative is there slightly modified to allow a possible jump in the normal derivative

$$(\partial_{\mathbf{n}} u^+ - \frac{\mu}{\mu_0}\partial_{\mathbf{n}} u^-)|_\Gamma = -\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma,$$

where μ_0 is the magnetic permeability of the medium and $\mu = \mu_p$ in Ω_p^- is the magnetic permeability of the domain Ω_p^-. The associated integral equation is then obtained by modifying the quantity

$$-\frac{I}{2} - N^-$$

by

$$\mu\left(-\frac{I}{2} - N^-\right)$$

in the matrix of (1.36).

1.4.3 Calderón projectors

Let us consider the Helmholtz representation of the interior field $u_T^- = u^-$ and the exterior one

$$\begin{cases} u_T^+ = u^{\text{inc}} - \mathscr{M}^+(u^+|_\Gamma) - \mathscr{L}^+(\partial_{\mathbf{n}} u^+|_\Gamma), \\ u_p^- = u_T^-|_{\Omega_p} = \mathscr{M}_p^-(u^-|_\Gamma) + \mathscr{L}_p^-(\partial_{\mathbf{n}} u^-|_\Gamma), \quad \forall p = 1, \ldots, M, \end{cases}$$

where again the $\pm$ upper index specifies the interior or exterior Green function. Note that the interior total field u_T^- is equal to the interior field u^-. However, in the propagation domain, the total field u_T^+ is obtained by summing the scattered wave u^+ and the incident wave u^{inc}. The idea is to obtain two sets of M equations, one taken from the interior and the other one from the exterior, and next to combine them. Let us first consider the M interior problems and take the trace and normal derivative trace of the interior fields (see Eq. (1.5) for the trace formulae)

$$\forall p = 1, \ldots, M, \qquad \begin{cases} u_p^-|_{\Gamma_p} = \left(\frac{I}{2} + M_p^-\right)(u_p^-|_{\Gamma_p}) + L_p^-(\partial_{\mathbf{n}} u_p^-|_{\Gamma_p}), \\ \partial_{\mathbf{n}} u_p^-|_{\Gamma_p} = D_p^-(u_p^-|_{\Gamma_p}) + \left(\frac{I}{2} + N_p^-\right)(\partial_{\mathbf{n}} u_p^-|_{\Gamma_p}). \end{cases}$$

Then, by introducing the following quantities

$$\forall p = 1, \ldots, M, \qquad \mathtt{U}_p^- = \begin{pmatrix} u_p^-|_{\Gamma_p} \\ \partial_{\mathbf{n}} u_p^-|_{\Gamma_p} \end{pmatrix},$$

the previous set of equations can be rewritten as

$$\forall p = 1, \ldots, M, \qquad \left(\mathbb{A}_{p,p}^- - \frac{I}{2}\right)\mathtt{U}_p^- = 0,$$

where

$$\mathbb{A}_{p,p}^- = \begin{pmatrix} M_{p,p}^- & L_{p,p}^- \\ D_{p,p}^- & N_{p,p}^- \end{pmatrix}.$$

The operator $L_{p,p}^-$ are the operators $L^{p,p}$ with the interior Green functions (same for the three others)

$$L_{p,p}^- \rho_p = \left(\int_{\Gamma_p} G^-(k\|\mathbf{x} - \mathbf{y}\|)\rho_p \, \mathrm{d}\Gamma_p\right)|_{\Gamma_p}.$$

Note that the Calderón projector $\mathbb{P}_p^-$, defined by

$$\mathbb{P}_p^- = \frac{I}{2} + \mathbb{A}_p,$$

is here hidden. This set of equations can be rewritten in the following matrix form

$$\begin{pmatrix} -\frac{I}{2} + \mathbb{A}_{1,1}^- & 0 & 0 & \dots & 0 \\ 0 & -\frac{I}{2} + \mathbb{A}_{2,2}^- & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -\frac{I}{2} + \mathbb{A}_{M,M}^- \end{pmatrix} \begin{pmatrix} \mathtt{U}_1^- \\ \mathtt{U}_2^- \\ \vdots \\ \mathtt{U}_M^- \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{1.37}$$

Now, a second set of equations can be obtained, thanks to the exterior part. Following the same idea, the trace and normal derivative trace of u_T^+ on Γ_p can be computed. However, the M contributions appear to be here

$$\forall p = 1, \dots, M, \qquad \begin{cases} u_T^+|_{\Gamma_p} = u^{\text{inc}}|_{\Gamma_p} + \dfrac{I}{2} u^+|_{\Gamma_p} - \displaystyle\sum_{q=1}^{M} \left[M_{p,q}^+(u^+|_{\Gamma_q}) + L_{p,q}^+(\partial_{\mathbf{n}} u^+|_{\Gamma_q}) \right], \\ \partial_{\mathbf{n}} u_T^+|_{\Gamma_p} = \partial_{\mathbf{n}} u^{\text{inc}}|_{\Gamma_p} + \dfrac{I}{2} \partial_{\mathbf{n}} u^+|_{\Gamma_p} - \left[D_{p,q}^+(u^+|_{\Gamma_q}) + N_{p,q}^+(\partial_{\mathbf{n}} u^+|_{\Gamma_q}) \right], \end{cases}$$

which can be rewritten as

$$\forall p = 1, \dots, M, \qquad \frac{I}{2} \mathtt{U}_p^+ + \sum_{q=1}^{M} \mathbb{A}_{p,q}^+ \mathtt{U}_q^+ = 0,$$

with the following quantities

$$\mathtt{U}_p^+ = \begin{pmatrix} u_T^+|_{\Gamma_p} \\ \partial_{\mathbf{n}} u_T^+|_{\Gamma_p} \end{pmatrix} \qquad \text{and} \qquad \mathbb{A}_{p,q}^+ = \begin{pmatrix} M_{p,q}^+ & L_{p,q}^+ \\ D_{p,q}^+ & N_{p,q}^+ \end{pmatrix}.$$

The operator $L_{p,q}^-$ is given by (same for the three others)

$$L_{p,q}^- \rho_q = \left(\int_{\Gamma_q} G^-(k\|\mathbf{x} - \mathbf{y}\|) \rho_q \, \mathrm{d}\Gamma_q \right) |_{\Gamma_p}.$$

Finally, this second set of equations can also be rewritten in the following matrix form

$$\begin{pmatrix} \frac{I}{2} + \mathbb{A}_{1,1}^+ & \mathbb{A}_{1,2}^+ & \mathbb{A}_{1,3}^+ & \dots & \mathbb{A}_{1,M}^+ \\ \mathbb{A}_{2,1}^+ & \frac{I}{2} + \mathbb{A}_{2,2}^+ & \mathbb{A}_{2,3}^+ & \dots & \mathbb{A}_{2,M}^+ \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbb{A}_{M,1}^+ & \mathbb{A}_{M,2}^+ & \mathbb{A}_{M,3}^+ & \dots & \frac{I}{2} + \mathbb{A}_{M,M}^+ \end{pmatrix} \begin{pmatrix} \mathtt{U}_1^+ - \mathtt{U}_1^{\text{inc}} \\ \mathtt{U}_2^+ - \mathtt{U}_2^{\text{inc}} \\ \vdots \\ \mathtt{U}_M^+ - \mathtt{U}_M^{\text{inc}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{1.38}$$

To obtain only one set of equations, systems (1.37) and (1.38) must be combined. A first-kind integral equation is obtained by doing "(1.37) + (1.38)", which is known to have a poor condition number but is robust, and a second-kind integral equation can be obtained by computing "(1.37) - (1.38)", which is well-conditioned but can provide inaccurate solution. These two integral equations are solved in the example function `BenchmarkCalderon`. The matrix $\mathbb{A}_{p,q}^{\pm}$ is computed by the `BlockCalderonProjector` function and the whole matrix $(\mathbb{A}_{p,q}^{\pm})_{p,q}$ by `CalderonProjector`.

Chapter 2

Multiple scattering by disks: approximation method in μ-diff

Contents

2.1 Spectral formulation used in μ-diff

We consider that the scatterers are some circular cylinders. In this case, we can explicitly compute the boundary integral equations in some Fourier bases, leading therefore to an efficient computational spectral method when used in conjunction with numerical linear algebra methods (direct or iterative solvers).

2.1.1 Notations and Fourier bases

Let us consider an orthonormal system $(\mathbf{O}, \overrightarrow{\mathbf{O}x_1}, \overrightarrow{\mathbf{O}x_2})$. We assume that the scattering obstacle Ω^- is the union of M disks Ω_p^-, for $p = 1, \ldots, M$, of radius a_p and center $\mathbf{O}_p$. We define Γ_p as the boundary of Ω_p^- and by $\Gamma = \cup_{p=1\ldots M}\Gamma_p$ the boundary of Ω^-. The unit normal vector $\mathbf{n}$ to Ω^- is outgoing. An illustration of the notations is reported in Figure 2.1.
For any $p = 1, \ldots, M$, we introduce $\mathbf{b}_p$ as the vector between the center $\mathbf{O}_p$ and the origin $\mathbf{O}$

$$\mathbf{b}_p = \mathbf{O}\mathbf{O}_p, \qquad b_p = \|\mathbf{b}_p\|, \qquad \alpha_p = Angle(\overrightarrow{\mathbf{O}x_1}, \mathbf{b}_p),$$

and, for $q = 1, \ldots, M$, with $q \neq p$, $\mathbf{b}_{pq}$ as the vector between the centers $\mathbf{O}_q$ and $\mathbf{O}_p$

$$\mathbf{b}_{pq} = \mathbf{O}_q\mathbf{O}_p, \qquad b_{pq} = \|\mathbf{b}_{pq}\|, \qquad \alpha_{pq} = Angle(\overrightarrow{\mathbf{O}x_1}, \mathbf{b}_{pq}).$$

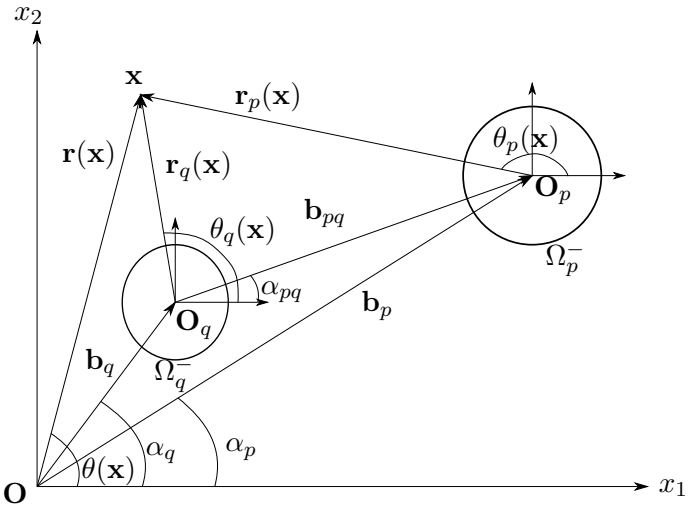


Figure 2.1: Illustration of the notations for two disks Ω_p^- and Ω_q^- and a point $\mathbf{x} \in \Omega^+$.

Furthermore, any point $\mathbf{x}$ is described by its global polar coordinates

$$\mathbf{r}(\mathbf{x}) = \mathbf{O}\mathbf{x}, \qquad r(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|, \qquad \theta(\mathbf{x}) = Angle(\overrightarrow{\mathbf{O}x_1}, \mathbf{r}(\mathbf{x})),$$

or by its polar coordinates in the orthonormal system associated with the obstacle Ω_p^-, with $p = 1, \ldots, M$,

$$\mathbf{r}_p(\mathbf{x}) = \mathbf{O}_p\mathbf{x}, \qquad r_p(\mathbf{x}) = \|\mathbf{r}_p(\mathbf{x})\|, \qquad \theta_p(\mathbf{x}) = Angle(\overrightarrow{\mathbf{O}_p x_1}, \mathbf{r}_p(\mathbf{x})).$$

Let us now build a basis of $L^2(\Gamma)$ to approximate the integral operators. To this end, we first construct a basis of $L^2(\Gamma_p)$ associated with Ω_p^-, for $p = 1, \ldots, M$. If the circle Γ_p has a radius one and is centered at the origin, then a suitable basis of $L^2(\Gamma_p)$ is the spectral Fourier basis of functions $(e^{im\theta})_{m\in\mathbb{Z}}$. We adapt this basis to the general case where $a_p \neq 1$ by introducing, on one hand, the functions $(\varphi_m)_{m\in\mathbb{Z}}$ defined on $\mathbb{R}^2$ by: $\forall m \in \mathbb{Z}$, $\forall \mathbf{x} \in \mathbb{R}^2$, $\varphi_m(\mathbf{x}) = e^{im\theta(\mathbf{x})}$, and, on the other hand, the functions $(\varphi_m^p)_{1\leqslant p\leqslant M,\, m\in\mathbb{Z}}$ given by

$$\forall p = 1, \ldots, M, \forall m \in \mathbb{Z}, \forall \mathbf{x} \in \Gamma_p, \qquad \varphi_m^p(\mathbf{x}) = \frac{\varphi_m(\mathbf{r}_p(\mathbf{x}))}{\sqrt{2\pi a_p}} = \frac{e^{im\theta_p(\mathbf{x})}}{\sqrt{2\pi a_p}}.$$

For $p = 1, \ldots, M$, the family $(\varphi_m^p)_{m\in\mathbb{Z}}$ forms an orthonormal basis of $L^2(\Gamma_p)$ for the hermitian inner product $(\cdot,\cdot)_{L^2(\Gamma_p)}$

$$\forall f, g \in L^2(\Gamma_p), \qquad (f,g)_{L^2(\Gamma_p)} = \int_{\Gamma_p} f(\mathbf{x})\overline{g(\mathbf{x})}\mathrm{d}\Gamma_p(\mathbf{x}).$$

To build a basis of $L^2(\Gamma)$, we introduce the functions Φ_m^p of $L^2(\Gamma)$ as the union of these M families

$$\forall p, q = 1, \ldots, M, \forall m \in \mathbb{Z}, \qquad \Phi_m^p|_{\Gamma_q} = \begin{cases} 0 & \text{if } q \neq p, \\ \varphi_m^p & \text{if } q = p. \end{cases}$$

The family $\mathscr{B} = \{\Phi_m^p,\, m \in \mathbb{Z}, p = 1, \ldots, M\}$, also called Fourier or spectral basis, is a Hilbert basis of $L^2(\Gamma)$ for the usual scalar product $(\cdot,\cdot)_{L^2(\Gamma)}$.

2.1.2 Integral operators - integral equations for a cluster of circular cylinders

In view of a numerical procedure, μ-diff can use for example the weak formulation of the EFIE (see Eq. 1.29, page 22) in $L^2(\Gamma)$ based on the Fourier basis $\mathscr{B}$

$$\begin{cases} \text{Find } \rho \in H^{-1/2}(\Gamma) \text{ such that for any } p = 1, \dots, M, \text{ and } m \in \mathbb{Z}, \\ (L\rho, \Phi_m^p)_{L^2(\Gamma)} = -\left(u^{\text{inc}}|_\Gamma, \Phi_m^p\right)_{L^2(\Gamma)}. \end{cases}$$

Since u^{inc} is assumed to be smooth enough (typically $\mathscr{C}^\infty$) and that Γ is $\mathscr{C}^\infty$, then the scattered wavefield is also $\mathscr{C}^\infty(\Omega^+)$ and the density ρ is (at least) in $H^{1/2}(\Gamma)$. Therefore, ρ can be developed in $\mathscr{B}$ as

$$\rho = \sum_{q=1}^{M} \sum_{n \in \mathbb{Z}} \rho_n^q \Phi_n^q$$

and the weak form of the EFIE is

$$\begin{cases} \text{Find the Fourier coefficients } \rho_n^q \in \mathbb{C}, \text{ for } q = 1, \dots, M, \text{ and } n \in \mathbb{Z}, \text{ such that,} \\ \forall p = 1, \dots, M, \ \forall m \in \mathbb{Z}, \qquad \displaystyle\sum_{q=1}^{M} \sum_{n \in \mathbb{Z}} \rho_n^q \left(L\Phi_n^q, \Phi_m^p\right)_{L^2(\Gamma)} = -\left(u^{\text{inc}}|_\Gamma, \Phi_m^p\right)_{L^2(\Gamma)}. \end{cases}$$

This formulation can be written under the following matrix form $\widetilde{\mathbb{L}}\widetilde{\boldsymbol{\rho}} = \widetilde{\mathbf{U}}$, where the infinite matrix representation $\widetilde{\mathbb{L}} = (\widetilde{\mathbb{L}}^{p,q})_{1 \leqslant p,q \leqslant M}$ and the infinite vectors $\widetilde{\boldsymbol{\rho}} = (\widetilde{\boldsymbol{\rho}}^p)_{1 \leqslant p \leqslant M}$ and $\widetilde{\mathbf{U}} = (\widetilde{\mathbf{U}}^p)_{1 \leqslant p \leqslant M}$ are defined by blocks as

$$\widetilde{\mathbb{L}} = \begin{bmatrix} \widetilde{\mathbb{L}}^{1,1} & \widetilde{\mathbb{L}}^{1,2} & \dots & \widetilde{\mathbb{L}}^{1,M} \\ \widetilde{\mathbb{L}}^{2,1} & \widetilde{\mathbb{L}}^{2,2} & \dots & \widetilde{\mathbb{L}}^{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\mathbb{L}}^{M,1} & \widetilde{\mathbb{L}}^{M,2} & \dots & \widetilde{\mathbb{L}}^{M,M} \end{bmatrix}, \qquad \widetilde{\boldsymbol{\rho}} = \begin{bmatrix} \widetilde{\boldsymbol{\rho}}^1 \\ \widetilde{\boldsymbol{\rho}}^2 \\ \vdots \\ \widetilde{\boldsymbol{\rho}}^M \end{bmatrix}, \qquad \widetilde{\mathbf{U}} = \begin{bmatrix} \widetilde{\mathbf{U}}^1 \\ \widetilde{\mathbf{U}}^2 \\ \vdots \\ \widetilde{\mathbf{U}}^M \end{bmatrix}, \tag{2.1}$$

with, for any $p, q = 1, \dots, M$, and $m, n \in \mathbb{Z}$: $\widetilde{\mathbb{L}}_{m,n}^{p,q} = (L\Phi_n^q, \Phi_m^p)_{L^2(\Gamma)}$, $\widetilde{\boldsymbol{\rho}}_m^p = \rho_m^p$ and $\widetilde{\mathbf{U}}_m^p = (-u^{\text{inc}}|_\Gamma, \Phi_m^p)_{L^2(\Gamma)}$.
For the other integral formulations (section 1.1.3) or even for any other boundary condition, the expressions of the three boundary integral operators M, N and D are needed. Therefore, to compute an integral equation, we introduce the infinite matrices

$$\widetilde{\mathbb{M}} = (\widetilde{\mathbb{M}}^{p,q})_{1 \leqslant p,q \leqslant M},$$

$$\widetilde{\mathbb{N}} = (\widetilde{\mathbb{N}}^{p,q})_{1 \leqslant p,q \leqslant M}$$

and

$$\widetilde{\mathbb{D}} = (\widetilde{\mathbb{D}}^{p,q})_{1 \leqslant p,q \leqslant M},$$

with the same block structure as $\widetilde{\mathbb{L}}$ (see equation (2.1)). For $p, q = 1, \dots, M$, the coefficients of the infinite matrices $\widetilde{\mathbb{M}}^{p,q}$, $\widetilde{\mathbb{N}}^{p,q}$ and $\widetilde{\mathbb{D}}^{p,q}$ are defined for any indices m and n in $\mathbb{Z}$ by

$$\widetilde{\mathbb{M}}_{m,n}^{p,q} = (M\Phi_n^q, \Phi_m^p)_{L^2(\Gamma)}, \widetilde{\mathbb{N}}_{m,n}^{p,q} = (N\Phi_n^q, \Phi_m^p)_{L^2(\Gamma)}, \text{ and } \widetilde{\mathbb{D}}_{m,n}^{p,q} = (D\Phi_n^q, \Phi_m^p)_{L^2(\Gamma)}.$$

For a numerical implementation, we can explicitly compute [6, 23] the matrix blocks $\widetilde{\mathbb{L}}^{p,q}$, $\widetilde{\mathbb{M}}^{p,q}$, $\widetilde{\mathbb{N}}^{p,q}$ and $\widetilde{\mathbb{D}}^{p,q}$ involved in $\widetilde{\mathbb{L}}$, $\widetilde{\mathbb{M}}$, $\widetilde{\mathbb{N}}$ and $\widetilde{\mathbb{D}}$, for $p, q = 1, \dots, M$. To this end, we introduce the infinite diagonal matrices $\widetilde{\mathbb{J}}^p$, $(\mathrm{d}\widetilde{\mathbb{J}})^p$, $\widetilde{\mathbb{H}}^p$ and $(\mathrm{d}\widetilde{\mathbb{H}})^p$, with general terms, for $m \in \mathbb{Z}$,

$$\widetilde{\mathbb{J}}_{mm}^p = J_m(ka_p), \quad (\mathrm{d}\widetilde{\mathbb{J}})_{mm}^p = J_m'(ka_p), \quad \widetilde{\mathbb{H}}_{mm}^p = H_m^{(1)}(ka_p), \quad (\mathrm{d}\widetilde{\mathbb{H}})_{mm}^p = H_m^{(1)'}(ka_p).$$

In addition, let $\widetilde{\mathbb{I}}^p$ be the infinite identity matrix, and, for $q \neq p$, the infinite separation matrix $\widetilde{\mathbb{S}}^{p,q}$ between the obstacles Ω_p^- and Ω_q^-, defined by

$$\widetilde{\mathbb{S}}^{p,q} = (\widetilde{\mathbb{S}}^{p,q}_{m,n})_{m\in\mathbb{Z},n\in\mathbb{Z}} \qquad \text{and} \qquad \widetilde{\mathbb{S}}^{p,q}_{m,n} = S_{mn}(\mathbf{b}_{pq}) = H^{(1)}_{m-n}(kb_{pq})e^{i(m-n)\alpha_{b_q}}.$$

Under these notations, we rewrite the blocks $\widetilde{\mathbb{L}}^{p,q}$, $\widetilde{\mathbb{M}}^{p,q}$, $\widetilde{\mathbb{N}}^{p,q}$ and $\widetilde{\mathbb{D}}^{p,q}$ of the infinite matrices $\widetilde{\mathbb{L}}$, $\widetilde{\mathbb{M}}$, $\widetilde{\mathbb{N}}$ and $\widetilde{\mathbb{D}}$ under the matrix form, for any $p, q = 1, \ldots, M$,

- $$\widetilde{\mathbb{L}}^{p,q} = \begin{cases} \dfrac{i\pi a_p}{2}\widetilde{\mathbb{J}}^p\widetilde{\mathbb{H}}^p, & \text{if } p = q, \\ \dfrac{i\pi\sqrt{a_p a_q}}{2}\widetilde{\mathbb{J}}^p(\widetilde{\mathbb{S}}^{p,q})^T\widetilde{\mathbb{J}}^q, & \text{if } p \neq q, \end{cases} \tag{2.2}$$

- $$\widetilde{\mathbb{M}}^{p,q} = \begin{cases} -\dfrac{1}{2}\widetilde{\mathbb{I}}^p - \dfrac{i\pi k a_p}{2}\widetilde{\mathbb{J}}^p(\mathrm{d}\widetilde{\mathbb{H}})^p = \dfrac{1}{2}\widetilde{\mathbb{I}}^p - \dfrac{i\pi k a_p}{2}(\mathrm{d}\widetilde{\mathbb{J}})^p\widetilde{\mathbb{H}}^p, & \text{if } p = q, \\ -\dfrac{ik\pi\sqrt{a_p a_q}}{2}\widetilde{\mathbb{J}}^p(\widetilde{\mathbb{S}}^{p,q})^T(\mathrm{d}\widetilde{\mathbb{J}})^q, & \text{if } p \neq q, \end{cases} \tag{2.3}$$

- $$\widetilde{\mathbb{N}}^{p,q} = \begin{cases} \dfrac{1}{2}\widetilde{\mathbb{I}}^p + \dfrac{i\pi k a_p}{2}\widetilde{\mathbb{J}}^p(\mathrm{d}\widetilde{\mathbb{H}})^p = -\dfrac{1}{2}\widetilde{\mathbb{I}}^p + \dfrac{i\pi k a_p}{2}(\mathrm{d}\widetilde{\mathbb{J}})^p\widetilde{\mathbb{H}}^p, & \text{if } p = q, \\ \dfrac{ik\pi\sqrt{a_p a_q}}{2}(\mathrm{d}\widetilde{\mathbb{J}})^p(\widetilde{\mathbb{S}}^{p,q})^T\widetilde{\mathbb{J}}^q, & \text{if } p \neq q, \end{cases} \tag{2.4}$$

- $$\widetilde{\mathbb{D}}^{p,q} = \begin{cases} \dfrac{i\pi k^2 a_p}{2}(\mathrm{d}\widetilde{\mathbb{J}})^p(\mathrm{d}\widetilde{\mathbb{H}})^p, & \text{if } p = q, \\ -\dfrac{ik^2\pi\sqrt{a_p a_q}}{2}(\mathrm{d}\widetilde{\mathbb{J}})^p(\widetilde{\mathbb{S}}^{p,q})^T(\mathrm{d}\widetilde{\mathbb{J}})^q, & \text{if } p \neq q, \end{cases} \tag{2.5}$$

where $(\widetilde{\mathbb{S}}^{p,q})^T$ is the transpose matrix of the separation matrix $\widetilde{\mathbb{S}}^{p,q}$.
The integral equations involve the trace or normal derivative trace of the incident wavefield on Γ. We have already introduced the infinite vector $\widetilde{\mathbf{U}}$ of the coefficients of $u^{\mathrm{inc}}|_\Gamma$ in the Fourier basis. We then define similarly the infinite vector $\mathrm{d}\widetilde{\mathbf{U}} = (\mathrm{d}\widetilde{\mathbf{U}}^p)_{1\leqslant p\leqslant M}$ of the coefficients of the normal derivative trace $\partial_{\mathbf{n}}u^{\mathrm{inc}}|_\Gamma$, such that

$$\forall p = 1, \ldots, M,\ \forall m \in \mathbb{Z}, \qquad (\mathrm{d}\widetilde{\mathbf{U}})^p_m = \left(\partial_{\mathbf{n}}u^{\mathrm{inc}}|_\Gamma, \Phi^p_m\right)_{L^2(\Gamma)}.$$

Finally, the density changes according to the integral equation and most particularly with respect to the boundary condition. To keep the same notations as previously, we introduce the densities λ and ψ (used in the BWIE) that are expanded in the Fourier basis as

$$\lambda = \sum_{p=1}^{M}\sum_{m\in\mathbb{Z}} \lambda^p_m\Phi^p_m \qquad \text{and} \qquad \psi = \sum_{p=1}^{M}\sum_{m\in\mathbb{Z}} \psi^p_m\Phi^p_m.$$

Finally, we set: $\widetilde{\boldsymbol{\lambda}} = (\widetilde{\boldsymbol{\lambda}}^p)_{1\leqslant p\leqslant M}$ and $\widetilde{\Psi} = (\widetilde{\Psi}^p)_{1\leqslant p\leqslant M}$, where each block $\widetilde{\boldsymbol{\lambda}}^p = (\widetilde{\boldsymbol{\lambda}}^p_m)_{m\in\mathbb{Z}}$ and $\widetilde{\Psi}^p = (\widetilde{\Psi}^p_m)_{m\in\mathbb{Z}}$ is defined by: $\forall m \in \mathbb{Z}$, $\widetilde{\boldsymbol{\lambda}}^p_m = \lambda^p_m$ and $\widetilde{\Psi}^p_m = \psi^p_m$.

2.1.3 Single-scattering preconditioned integral equations

The EFIE preconditioned by its single scattering component (see Section 1.2.2), given by

$$\widehat{L}^{-1}L\rho = \widehat{L}^{-1}u^{\text{inc}}|_{\Gamma},$$

can also be computed analytically in the Fourier bases. Indeed, let $\widehat{\mathbb{L}}^{-1}\mathbb{L}$ be the matrix associated to the operator $\widehat{L}^{-1}L$, then

$$\forall p,q = 1,\dots,M, \qquad (\widehat{\widetilde{\mathbb{L}}}^{-1}\widetilde{\mathbb{L}})^{p,q} = \begin{cases} \mathbb{I}^p & \text{if } p = q, \\ \sqrt{\dfrac{a_q}{a_p}}(\widetilde{\mathbb{H}}^p)^{-1}(\widetilde{\mathbb{S}}^{p,q})^T\widetilde{\mathbb{J}}^q & \text{otherwise.} \end{cases} \tag{2.6}$$

For the Neumann case, the preconditioned integral equation (EFIE) reads as

$$\widehat{D}^{-1}D\lambda = \widehat{D}^{-1}\partial_{\mathbf{n}}u^{\text{inc}}|_{\Gamma}.$$

The matrix representation $\widehat{\mathbb{D}}^{-1}\mathbb{D}$ of $\widehat{D}^{-1}D$ is then given by

$$\forall p,q = 1,\dots,M, \qquad (\widehat{\widetilde{\mathbb{D}}}^{-1}\widetilde{\mathbb{D}})^{p,q} = \begin{cases} \mathbb{I}^p & \text{if } p = q, \\ -\sqrt{\dfrac{a_q}{a_p}}((\mathrm{d}\widetilde{\mathbb{H}})^p)^{-1}(\widetilde{\mathbb{S}}^{p,q})^T(\mathrm{d}\widetilde{\mathbb{J}})^q & \text{otherwise.} \end{cases} \tag{2.7}$$

As highlighted by Proposition 1.9, there is no need to compute the preconditioned versions of the other integral equations as they lead to the same operator (up to an invertible operators, for BWIE). To solve sound-hard or sound-soft scattering, using the above integral equations appears to be a suitable choice.

2.1.4 Projection of the incident waves in the Fourier basis

To fully solve one of the integral equations (EFIE, MFIE, CFIE or BWIE), we need to compute the Fourier coefficients of the trace and normal derivative traces of the incident wave. We give the results for both an incident plane wave and a pointwise source term (Green's function). For an incident plane wave, the following proposition holds [2].

Proposition 2.1. *Let us assume that u^{inc} is an incident plane wave of direction $\boldsymbol{\beta}$, with $\boldsymbol{\beta} = (\cos(\beta), \sin(\beta))$ and $\beta \in [0, 2\pi]$, i.e.*

$$\forall \mathbf{x} \in \mathbb{R}^2, \qquad u^{inc}(\mathbf{x}) = e^{ik\boldsymbol{\beta}\cdot\mathbf{x}}.$$

Then we have the following equalities

$$\widetilde{\mathbf{U}}^p_m = \left(u^{inc}|_{\Gamma}, \Phi^p_m\right)_{L^2(\Gamma)} = d^p_m J_m(ka_p), \qquad (\mathrm{d}\widetilde{\mathbf{U}})^p_m = \left(\partial_{\mathbf{n}}u^{inc}|_{\Gamma}, \Phi^p_m\right)_{L^2(\Gamma)} = kd^p_m J'_m(ka_p),$$

with $d^p_m = \sqrt{2\pi a_p}e^{ik\boldsymbol{\beta}\cdot\mathbf{b}_p}e^{im(\pi/2-\beta)}$.

Let us consider now an incident wave emitted by a pointwise source located at $\mathbf{s} \in \Omega^+$, i.e. the wave u^{inc} is the Green's function centered at $\mathbf{s}$. The Fourier coefficients of the trace and normal derivative trace of u^{inc} on Γ are then given by the following proposition [23].

Proposition 2.2. *Let $\mathbf{s} \in \Omega^+$. We assume that the incident wave u^{inc} is the Green's function centered at $\mathbf{s}$*

$$\forall \mathbf{x} \in \mathbb{R}^2 \setminus \{\mathbf{s}\}, \qquad u^{inc}(\mathbf{x}) = G(\mathbf{x}, \mathbf{s}) = \frac{i}{4} H_0^{(1)}(k\|\mathbf{x} - \mathbf{s}\|).$$

The Fourier coefficients in $\mathscr{B}$ of the trace and normal derivative trace of the incident wave on Γ are respectively given by

$$\widetilde{\mathbf{U}}_m^p = \left(u^{inc}|_\Gamma, \Phi_m^p\right)_{L^2(\Gamma)} = \frac{i\pi a_p}{2} J_m(ka_p) H_m^{(1)}(kr_p(\mathbf{s})) \overline{\widetilde{\Phi}_m^p(\mathbf{s})}$$

and

$$(\mathrm{d}\widetilde{\mathbf{U}})_m^p = \left(\partial_{\mathbf{n}} u^{inc}|_\Gamma, \Phi_m^p\right)_{L^2(\Gamma)} = k \frac{i\pi a_p}{2} J_m'(ka_p) H_m^{(1)}(kr_p(\mathbf{s})) \overline{\widetilde{\Phi}_m^p(\mathbf{s})}.$$

2.1.5 Near-field evaluation

Outside the obstacles

By using the Graf's addition theorem [18, 23], we can compute the expression of the single- and double-layer potentials at a point $\mathbf{x}$ located in the propagation domain Ω^+.

Proposition 2.3. *Let $\rho \in L^2(\Gamma)$ and $\mu \in H^{1/2}(\Gamma)$ be two densities admitting the following decompositions in the Fourier basis $\mathscr{B}$*

$$\rho = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \rho_m^p \Phi_m^p \qquad \text{and} \qquad \lambda = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \lambda_m^p \Phi_m^p.$$

Then, for any point $\mathbf{x}$ in the domain of propagation Ω^+, the single-layer potential reads

$$\mathscr{L}\rho(\mathbf{x}) = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \rho_m^p \mathscr{L}\Phi_m^p(\mathbf{x}) = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \rho_m^p \frac{i\pi a_p}{2} J_m(ka_p) H_m^{(1)}(kr_p(\mathbf{x})) \widetilde{\Phi}_m^p(\mathbf{x}), \tag{2.8}$$

and the double-layer potential can be expressed as

$$\mathscr{M}\lambda(\mathbf{x}) = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \lambda_m^p \mathscr{M}\Phi_m^p(\mathbf{x}) = -\sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \lambda_m^p \frac{i\pi k a_p}{2} J_m'(ka_p) H_m^{(1)}(kr_p(\mathbf{x})) \widetilde{\Phi}_m^p(\mathbf{x}). \tag{2.9}$$

Proposition 2.3 implies that, for any $\mathbf{x}$ in Ω^+,

$$u(\mathbf{x}) = \mathscr{L}\rho(\mathbf{x}) + \mathscr{M}\lambda(\mathbf{x}) = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \frac{i\pi a_p}{2} \left[\rho_m^p J_m(ka_p) + \lambda_m^p J_m'(ka_p)\right] H_m^{(1)}(kr_p(\mathbf{x})) \widetilde{\Phi}_m^p(\mathbf{x}).$$

Inside the obstacles

Similarly, the potentials can be computed inside the obstacles, which is useful for penetrable obstacles for instance. In this case, only the contribution of the current obstacle is taken into account

$$u^-(\mathbf{x}) = \mathscr{L}_p \rho_p + \mathscr{M}_p \lambda_p, \qquad \forall \mathbf{x} \in \Omega_p.$$

Proposition 2.4. *Let $\rho \in L^2(\Gamma)$ and $\mu \in H^{1/2}(\Gamma)$ be two densities admitting the following decompositions in the Fourier basis $\mathscr{B}$*

$$\rho = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \rho_m^p \Phi_m^p \qquad \text{and} \qquad \lambda = \sum_{p=1}^{M} \sum_{m \in \mathbb{Z}} \lambda_m^p \Phi_m^p.$$

Then, for any point $\mathbf{x}$ *inside the obstacle* Ω_p*, the single-layer potential reads*

$$\mathscr{L}\rho(\mathbf{x}) = \sum_{m\in\mathbb{Z}} \rho_m^p \mathscr{L}\Phi_m^p(\mathbf{x}) = \sum_{p=1}^{M}\sum_{m\in\mathbb{Z}} \rho_m^p \frac{i\pi a_p}{2} H_m^{(1)}(ka_p) J_m(kr_p(\mathbf{x}))\widetilde{\Phi}_m^p(\mathbf{x}), \tag{2.10}$$

and the double-layer potential can be expressed as

$$\mathscr{M}\lambda(\mathbf{x}) = \sum_{p=1}^{M}\sum_{m\in\mathbb{Z}} \lambda_m^p \mathscr{M}\Phi_m^p(\mathbf{x}) = -\sum_{p=1}^{M}\sum_{m\in\mathbb{Z}} \lambda_m^p \frac{i\pi k a_p}{2} H_m^{(1)'}(ka_p) J_m(kr_p(\mathbf{x}))\widetilde{\Phi}_m^p(\mathbf{x}). \tag{2.11}$$

2.1.6 Far-field and Radar Cross Section (RCS)

For computing the far-field pattern, let us recall that the scattered field u admits the following Helmholtz's integral representation: $u = \mathscr{L}\rho + \mathscr{M}\lambda$, where ρ and λ are two unknown densities. In the polar coordinates system (r,θ) and by using an asymptotic expansion of u when $r \to +\infty$, the following relation holds [14]

$$\forall\theta\in[0,2\pi], \qquad u(r,\theta) = \frac{e^{ikr}}{r^{1/2}}\left[a_{\mathscr{L}}(\theta) + a_{\mathscr{M}}(\theta)\right] + O\left(\frac{1}{r^{3/2}}\right),$$

where $a_{\mathscr{L}}$ and $a_{\mathscr{M}}$ are the radiated far-fields for the single- and double-layer potentials, respectively, defined for any angle θ of $[0,2\pi]$ by

$$\begin{cases} a_{\mathscr{L}}(\theta) = \dfrac{1}{\sqrt{8k\pi}} e^{i\pi/4} \displaystyle\int_\Gamma e^{-ik\boldsymbol{\theta}\cdot\mathbf{y}} \rho(\mathbf{y}) \mathrm{d}\Gamma(\mathbf{y}), \\ a_{\mathscr{M}}(\theta) = \dfrac{1}{\sqrt{8k\pi}} e^{i\pi/4} \displaystyle\int_\Gamma -\frac{ik}{\|\mathbf{y}\|}\boldsymbol{\theta}\cdot\mathbf{y} e^{-ik\boldsymbol{\theta}\cdot\mathbf{y}} \lambda(\mathbf{y}) \mathrm{d}\Gamma(\mathbf{y}), \end{cases}$$

with $\boldsymbol{\theta} := (\cos(\theta), \sin(\theta))$. In addition, the Radar Cross Section (RCS) is defined by

$$\forall\theta\in[0,2\pi], \quad \mathrm{RCS}(\theta) = 10\log_{10}\left(2\pi\left|a_{\mathscr{L}}(\theta) + a_{\mathscr{M}}(\theta)\right|^2\right)(\mathrm{dB}).$$

To optimize the far-fields computation, these relations can be written thanks to the inner product between two infinite vectors. Indeed, let us introduce $\widetilde{\mathbf{a}}_{\mathscr{L}} = ((\widetilde{\mathbf{a}}_{\mathscr{L}})^p)_{1\leqslant p\leqslant M}$ and $\widetilde{\mathbf{a}}_{\mathscr{M}} = ((\widetilde{\mathbf{a}}_{\mathscr{M}})^p)_{1\leqslant p\leqslant M}$, where $(\widetilde{\mathbf{a}}_{\mathscr{L}})^p$ and $(\widetilde{\mathbf{a}}_{\mathscr{M}})^p$ are given by: $\forall p = 1,\ldots,M$,

$$\begin{cases} (\widetilde{\mathbf{a}}_{\mathscr{L}})^p = \left((\widetilde{\mathbf{a}}_{\mathscr{L}})_m^p\right)_{m\in\mathbb{Z}}, & (\widetilde{\mathbf{a}}_{\mathscr{L}})_m^p = \dfrac{ie^{-i\pi/4}\sqrt{a_p}}{2\sqrt{k}} e^{-ib_p k\cos(\theta-\alpha_p)} J_m(ka_p) e^{im(\theta-\pi/2)}, \\ (\widetilde{\mathbf{a}}_{\mathscr{M}})^p = \left((\widetilde{\mathbf{a}}_{\mathscr{L}})_m^p\right)_{m\in\mathbb{Z}}, & (\widetilde{\mathbf{a}}_{\mathscr{L}})_m^p = \dfrac{ie^{-i\pi/4}\sqrt{ka_p}}{2} e^{-ib_p k\cos(\theta-\alpha_p)} J_m'(ka_p) e^{im(\theta-\pi/2)}. \end{cases} \tag{2.12}$$

Then, we obtain the following: $a_{\mathscr{L}}(\theta) = (\widetilde{\mathbf{a}}_{\mathscr{L}})^T\widetilde{\boldsymbol{\rho}}$ and $a_{\mathscr{M}}(\theta) = (\widetilde{\mathbf{a}}_{\mathscr{M}})^T\widetilde{\boldsymbol{\lambda}}$.

2.2 Finite-dimensional approximations and numerical solutions proposed in μ-diff

We now have all the ingredients to numerically solve the four integral equations EFIE, MFIE, CFIE and BWIE, for sound-soft obstacles. In fact, any integral equation for any boundary condition can be solved according to the previous developments. In practice, the resulting infinite Fourier system needs to be truncated to get a finite dimensional problem: we must pass

from a sum over $m \in \mathbb{Z}$ to a finite number of Fourier modes that depends on ka_p, $p = 1, ..., M$. Let us consider e.g. the EFIE, the extension to the other boundary integral operators being direct. The EFIE is given by equation (2.1): $\widetilde{\mathbb{L}}\widetilde{\boldsymbol{\rho}} = -\widetilde{\mathbf{U}}$. To truncate each Fourier series associated with $(\Phi_m^p)_{m\in\mathbb{Z}}$ for the obstacle Ω_p^-, we only keep $2N_p + 1$ modes in such a way that the indices m of the truncated series satisfy: $\forall p = 1, \ldots, M$, $-N_p \leqslant m \leqslant N_p$. The truncation parameter N_p must be fixed large enough, with $N_p \geqslant ka_p$, for $p = 1, ..., M$. An example [2, 6] is: $N_p = ka_p + C_p$, where C_p weakly grows with ka_p. A numerical study of the parameter N_p is proposed in [2, 6] where the following formula leads to a stable and accurate computation

$$N_p = \left[ka_p + \left(\frac{1}{2\sqrt{2}} \ln(2\sqrt{2}\pi ka_p \varepsilon^{-1}) \right)^{\frac{2}{3}} (ka_p)^{1/3} + 1 \right], \tag{2.13}$$

where ε is a small parameter (related to the relative tolerance required in the iterative Krylov subspace solver used for solving the truncated linear system (2.14), see [2, 6]).
The resulting linear system writes

$$\mathbb{L}\boldsymbol{\rho} = -\mathbf{U}, \tag{2.14}$$

where we introduced the block matrix $\mathbb{L} = (\mathbb{L}^{p,q})_{1\leqslant p,q\leqslant M}$, the vectors $\boldsymbol{\rho} = (\boldsymbol{\rho}^p)_{1\leqslant p\leqslant M}$ and $\mathbf{U} = (\mathbf{U}^p)_{1\leqslant p\leqslant M}$ as

$$\mathbb{L} = \begin{bmatrix} \mathbb{L}^{1,1} & \mathbb{L}^{1,2} & \ldots & \mathbb{L}^{1,M} \\ \mathbb{L}^{2,1} & \mathbb{L}^{2,2} & \ldots & \mathbb{L}^{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{L}^{M,1} & \mathbb{L}^{M,2} & \ldots & \mathbb{L}^{M,M} \end{bmatrix}, \qquad \boldsymbol{\rho} = \begin{bmatrix} \boldsymbol{\rho}^1 \\ \boldsymbol{\rho}^2 \\ \vdots \\ \boldsymbol{\rho}^M \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} \mathbf{U}^1 \\ \mathbf{U}^2 \\ \vdots \\ \mathbf{U}^M \end{bmatrix}. \tag{2.15}$$

For $p, q = 1, \ldots, M$, the complex-valued matrix $\mathbb{L}^{p,q}$ is of size $(2N_p + 1) \times (2N_q + 1)$ and its coefficients $\mathbb{L}^{p,q}_{m,n}$ are: $\mathbb{L}^{p,q}_{m,n} = \widetilde{\mathbb{L}}^{p,q}_{m,n}$, for $m = -N_p, \ldots, N_p$, $n = -N_q, \ldots, N_q$. The complex-valued components of the vector $\boldsymbol{\rho}^p = (\boldsymbol{\rho}^p_m)_{-N_p\leqslant m\leqslant N_p}$ of size $2N_p + 1$ are the approximate Fourier coefficients ρ^p_m of ρ. For the sake of clarity, we keep on writing: $\boldsymbol{\rho}^p_m = \widetilde{\boldsymbol{\rho}}^p_m = \rho^p_m$, for all $m = -N_p, \ldots, N_p$. The complex-valued vector $\mathbf{U}^p = (\mathbf{U}^p_m)_{-N_p\leqslant m\leqslant N_p}$ is composed of the $2N_p+1$ Fourier coefficients of the trace of the incident wave on Γ, i.e. $\mathbf{U}^p_m = \widetilde{\mathbf{U}}^p_m = (u^{\text{inc}}|_\Gamma, \Phi^p_m)_{L^2(\Gamma)}$, $\forall m = -N_p, \ldots, N_p$. If $N_{\text{tot}} = \sum_{p=1}^M (2N_p + 1)$ denotes the total number of modes, the size of the complex-valued matrix $\mathbb{L}$ is then $N_{\text{tot}} \times N_{\text{tot}}$. More generally, all the boundary integral operators can be truncated according to this process. Concerning the notations, it is sufficient to formally omit the tilde symbol $\sim$ over the quantities involved in sections (2.1.2) to (2.1.6).
Since the four finite-dimensional matrices $\mathbb{L}$, $\mathbb{M}$, $\mathbb{N}$ and $\mathbb{D}$ that respectively correspond to the four boundary integral operators L, M, N and D can be computed, the linear systems that approximate the EFIE, MFIE, CFIE and BWIE can be stated. For example, the CFIE leads to (with $0 \leqslant \alpha \leqslant 1$ and $\Im(\eta) \neq 0$)

$$\left[\alpha\eta\mathbb{L} + (1-\alpha)\left(\frac{\mathbb{I}}{2} + \mathbb{N} \right) \right] \boldsymbol{\rho} = -\alpha\eta\mathbf{U} - (1-\alpha)\mathrm{d}\mathbf{U}. \tag{2.16}$$

Let us remark that the matrix obtained after discretization is always a linear combination of the four integral operators $\mathbb{L}$, $\mathbb{M}$, $\mathbb{N}$, $\mathbb{D}$ and the identity matrix $\mathbb{I}$. As a consequence, for a given integral equation, the resulting matrix is of size $N_{\text{tot}} \times N_{\text{tot}}$ and has the same block structure as e.g. $\mathbb{L}$ (see equation (2.15)). The finite-dimensional linear system (2.14) (or (2.16)) is accurately solved in μ-diff by using the Matlab direct solver or a preconditioned Krylov subspace linear solver that uses fast matrix-vector products based on Fast Fourier Transforms (FFTs), the choice of the linear algebra strategy (direct vs. iterative) depending on the configuration with respect to ka_p and M. The use of FFTs is made possible since the off-diagonal blocks of the

integral operators can be written as the products of diagonal and Toeplitz matrices [2, 6] (see e.g. the matrices $\widetilde{\mathbb{S}}_{m,n}^{p,q}$ in section 2.1.2). In addition, low memory is only necessary when ka_p is large enough since the storage of the Toeplitz matrices can be optimized. This resulting storage technique is called *sparse* representation in μ-diff, in contrast with the *dense* (full) storage of the complex-valued matrices. Let us assume that $a_p \approx a$, for $1 \leqslant p \leqslant M$. In terms of storage, the dense version of a matrix requires to store about $4M^2[ka]^2$ coefficients (assuming that N_p are fixed by formula (2.13), and $[r]$ denotes the integer part of a real number r) while the sparse storage needs about $4M^2[ka]$ complex-valued coefficients. In terms of computational time for solving the linear system, the direct (multi-threaded) gaussian solver included in Matlab leads to a cost that scales with $\mathcal{O}(M^3(ka)^3)$. For the preconditioned iterative Krylov subspace methods (i.e. restarted GMRES)), the global cost is $\mathcal{O}(M^2 ka \log_2(ka))$, the converge rate depending on the physical situation and robustness of the preconditioner. From these remarks, we deduce that an iterative method can be an efficient alternative to a direct solver for large wavenumbers ka, but also for large M. We refer to [2, 6] for a thorough computational study of the various numerical strategies. A few examples in μ-diff are provided with the toolbox. Finally, the post-processing formulas (near- and far-fields quantities) clearly inherits of the truncation procedure (see sections 2.1.5 and 2.1.6).

Chapter 3

Description of the μ-diff toolbox and first examples

Contents

3.1 Generalities

An effort has been made to keep the same notations between the mathematical framework and the μ-diff toolbox. Some explanations (goal of the function and input/output arguments) about a function `name_of_the_function` can be obtained by typing the classical Matlab command window

```
help name_of_the_function;
```

Because μ-diff includes all the integral operators that are needed in scattering (traces and normal derivative traces of the single- and double-layer potentials), a large class of scattering problems can be solved as long as the shape of the obstacles is circular. Concerning the geometrical configurations, any deterministic or random distribution of disks is possible. Finally, μ-diff includes post-processing facilities like e.g.: surface and far-fields computations, total and scattered exterior (near-field) visualization...

We now introduce the μ-diff toolbox based on Matlab by explaining the main predefined functions and their relations with the previous mathematical derivations. Section 3.2 gives a presentation and introduces the notations. Section 3.3 shows how to define the scattering configuration (geometry and physical parameters). Section 3.4 presents the way the integral equations must be defined and solved. Finally, section 3.5 describes the data post-processing.
The μ-diff toolbox is organized following the five subdirectories (see also Figure 3.1 for a diagram of the directory arborescence)

- `mudiff/PreProcessing/`: pre-processing data functions (incident wave and geometry) (section 3.3).
- `mudiff/IntOperators/`: functions for the four basic integral operators (dense and sparse structure) used in the definition of the integral equations to solve (section 3.4).
- `mudiff/PostProcessing/`: post-processing functions of the solution (trace and normal derivative traces, computation of the scattered/total wavefield at some points of the spatial domain or on a grid, far-field and RCS) (section 3.5).
- `mudiff/Common/`: this directory includes functions that are used in μ-diff but which does not need to be known from the standard user point of view.
- `mudiff/Examples/`: various scripts are presented for the user in standard configurations.

In addition, the μ-diff user guide, license and credits can be found under the directory `mudiff/Doc/`.

3.2 Common argument and notations

A large number of arguments of the μ-diff functions are similar. For the sake of conciseness and if nothing is specified, the following arguments refer to the ones given in the Tables below, where the indices p and q vary from 1 to M (or `N_scat` in the μ-diff scripts). In addition, any function or value specific to μ-diff is written with the following `font`.

Geometry

Name	Size	Description
`N_scat`	$[1 \times 1]$	number of obstacles M
`O`	$[2 \times M]$	matrix of the centers of the disks such that `O(1,p)` (resp. `O(2,p)`) is the abscissa (resp. ordinate) of the p^{th} obstacle
`Op`	$[2 \times 1]$	coordinates of the p^{th} scatterer
`Oq`	$[2 \times 1]$	coordinates of the q^{th} scatterer
`a`	$[1 \times M]$	vector of the radii of the disks such that `a(p)` is the radius of the p^{th} scatterer
`ap`	$[1 \times 1]$	radius of the p^{th} scatterer
`aq`	$[1 \times 1]$	radius of the q^{th} scatterer

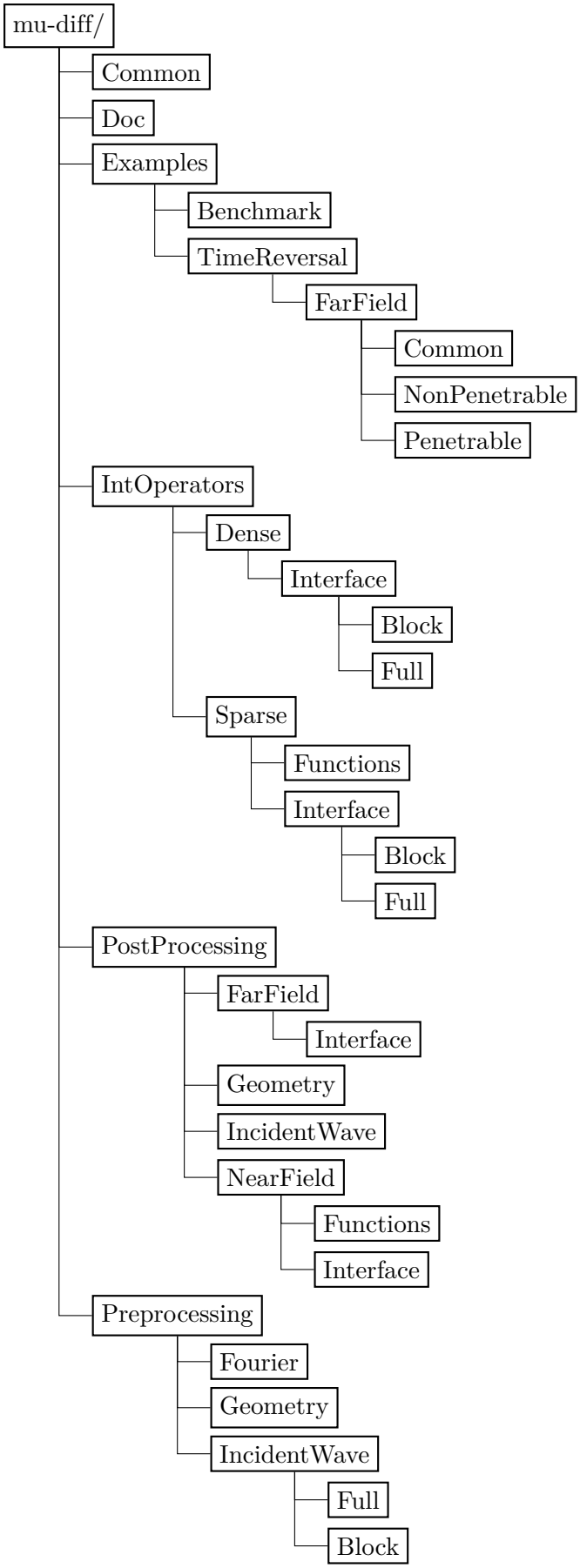


Figure 3.1: Arborescence of μ-diff toolbox.

Parameters (wavenumbers, incident waves, Fourier series expansions, ...)

Name	Size	Description
`beta_inc`	$[1 \times 1]$	angle of direction β of a plane wave $e^{ik(\cos(\beta)x_1+\sin(\beta)x_2)}$
`XS`	$[2 \times 1]$	center (x_{1s}, x_{2s}) of a point source: x_{1s} =`XS(1)` and x_{2s} =`XS(2)`. A point source wave is given by $$\frac{i}{4}H_0^{(1)}(k\|\mathbf{x}-\mathbf{x}_s\|),$$ with $\mathbf{x}=(x_1,x_2)$ and $\mathbf{x}_s=(x_{1s},x_{2s})$, with $H_0^{(1)}$ the zeroth order Hankel function of the first-kind
`k`	$[1 \times 1]$	wavenumber k in the vacuum
`k_int`	$[1 \times M]$	wavenumbers in the obstacles: k_p^- =`k_int(p)`. If `k_int` is a scalar then k_p^- = `k_int` for all $p = 1, ..., M$
`M_modes`	$[1 \times M]$	vector of index of truncation of the Fourier series, *i.e.* `M_modes(p)`$=N_p$
`Np`	$[1 \times 1]$	corresponds to the truncation index N_p in the Fourier series
`Nq`	$[1 \times 1]$	corresponds to the truncation index N_q in the Fourier series

Integral operators (see chapter 1)

Index	Letter	μ-diff abbreviation	Operator
0	-	-	null operator
1	I	`Identity`	identity
2	L	`SingleLayer`	$L\rho = \int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\,\mathrm{d}\mathbf{y}$
3	M	`DoubleLayer`	$M\lambda = -\int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\,\mathrm{d}\mathbf{y}$
4	N	`DnSingleLayer`	$N\rho = \partial_{\mathbf{n_x}} \int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\,\mathrm{d}\mathbf{y}$
5	D	`DnDoubleLayer`	$D\lambda = -\partial_{\mathbf{n_x}} \int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\,\mathrm{d}\mathbf{y}$
6	$\hat{L}^{-1}L$	`PrecondDirichlet`	single-scattering preconditioned trace of the single-layer operator (see §1.2.2)
7	$\hat{D}^{-1}D$	`PrecondNeumann`	single-scattering preconditioned normal derivative trace of the double-layer operator (see §1.2.2)

3.3 Pre-processing

The pre-processing in μ-diff consists in defining the right-hand side (or the incident wave) and the geometry (the obstacles). The associated functions are located respectively in the folders `mudiff/PreProcessing/IncidentWave` and in `mudiff/PreProcessing/Geometry`.

3.3.1 Geometry: creating the obstacles

The obstacles are stored in memory as a row vector `a` containing the radii of the disks and a $[2 \times$ `N_scat`$]$ matrix `O` with the centers of the disks (`O(1,p=`x- and `O(2,p=`y- coordinates of the center of the p^{th} disk). These two arrays can be created either manually or by using some built-in functions.

Manually

The disks can be created manually by simply creating the two variables `O` and `a` containing respectively the coordinates of the disks and their radii. For example, for three obstacles placed at points $(-1, 2)$, $(5, 5)$ and $(-15, 10)$ with respective radii 0.1, 0.5 and 10, one gets

```
O = [-1, 5, 2 ; -15, 5, 10];
a = [0.1, 0.5, 10];
```

Periodic placement

Two built-in functions are available with the toolbox to periodically place some disks, with a rectangular or a triangular lattice, as shown on Figure 3.2. The two functions must be called as follows, first for the rectangular lattice

```
O = RectangularLattice(bx, by, Nx, Ny, OPTIONS);
```

and second for the triangular lattice

```
O = TriangularLattice(bx, by, Nx, Ny, OPTIONS);
```

where the arguments are

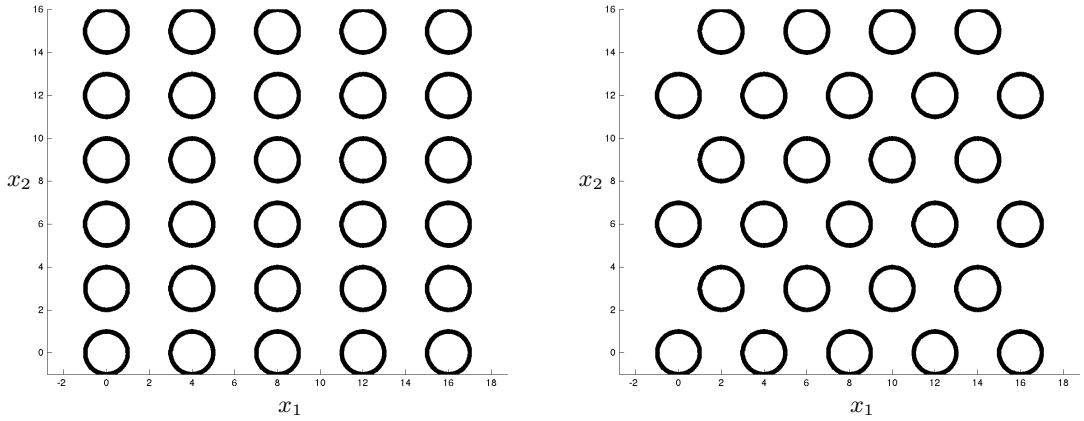
- `bx`: distance separating two centers in the x_1-direction (be careful with the radii of the disks to avoid overlapping!).
- `by`: distance separating two row of obstacles in the x_2-direction.
- `Nx`: number of disks in a row.
- `Ny`: number of rows.

For both functions, the vector of radii must be built separately and manually. For a set of unitary disks, the following command is used

```
a = ones(size(O,2));
```

The available options are (same for `RectangularLattice` and `TriangularLattice`)

- `RectangularLattice(..., 'Origin', Ostart)` places the first disk at `Ostart` position, `Ostart` being a [2x1] vector. (Default: $[0; 0]$).
- `RectangularLattice(..., 'Centered', Ocenter)` center the collection on the point `Ocenter`, `Ocenter` being a [2x1] vector. (Default: none but replace Ostart value).
- `RectangularLattice(..., 'Direction', DIR)` places the row in the increasing (`DIR` = $+1$) or decreasing x_2-direction (`DIR` = -1). The default value is: `DIR` = $+1$.



(a) Rectangular lattice (b) Triangular lattice

Figure 3.2: Rectangular (left) and triangular (right) lattice with `bx` $= 3$, `by` $= 4$, `Nx` $= 5$, `Ny`$=6$.

Example: Building a rectangular lattice of 3×4 unit disks. Each disk is separated from the other by a distance equal to 1.5 (so the centers are separated from a distance of 3.5)

```
O = RectangularLattice(3.5, 3.5, 3, 4);
a = ones(size(O,2));
```

Random placement

The toolbox μ-diff provides a function `CreateRandomDisks` to randomly place `N_scat` obstacles in a box $[$`xmin`, `xmax`$] \times [$`ymin`, `ymax`$]$ with a random radius. In its simplest version, the function is called as

```
[O, a] = CreateRandomDisks(xmin, xmax, ymin, ymax, N_scat);
```

In that case, `CreateRandomDisks` builds `N_scat` disks with unit radius in the box. The function takes care to not overlap the disks. Note that it is possible that the function does not succeed to place the obstacles (e.g. if the user specifies too many obstacles in a box which is not large enough). This is the reason why a security test has been set: only 500 possible placements are allowed per disk.

The function comes along with a large set of optional arguments

```
[O, a] = CreateRandomDisks(xmin, xmax, ymin, ymax, N_scat,
           amin, amax, dmim, dmax, O_avoid, a_avoid, dmin_avoid, dmax_avoid);
```

where each additional argument is optional (but the order must be kept (`amin` must be set, then `amax`, etc...!) and given by

Variable	Type	Default	Description
amin	scalar	1	minimal (random) radius of the obstacles allowed
amax	scalar	1	maximal (random) radius of the obstacles allowed
dmin	scalar	realmin	minimal distance allowed between two obstacles (not between the centers!). Setting ≤ 0 value will set dmin to realmin (*i.e.* ignore it)
dmax	scalar	realmax	maximal distance allowed between two obstacles (not between the centers!). The maximal distance is efficiently reached! Setting ≤ 0 value will set dmax to realmax (i.e. ignore it)
O_avoid	$[2 \times N]$	[]	center of N hole(s) where the obstacles must not overlap. Useful for example for the points source location
a_avoid	$[1 \times N]$	[]	radii of the N holes
dmin_avoid	$[1 \times N]$	[]	minimal distance between an obstacle and a hole

The "holes", represented by the *_avoid arguments, are circular regions where the obstacles must not overlap, for example where a point source is emitting a wave.

Example 1: creating N_scat random disks with random radii is realized by the command

```
[O, a] = CreateRandomDisks(xmin, xmax, ymin, ymax, N_scat, amin, amax);
```

Example 2: building 7 obstacles in the box $[-10, 10] \times [-10, 10]$ with radii between 0.1 and 0.5. The disks must be separated by a minimal distance equal to 0.1 and without maximal value. The command is then

```
[O, a] = CreateRandomDisks(-10, 10, -10, 10, 7, 0.1, 0.5, 0.1, -1);
```

Example 3: now consider that a point source is located at $(2, 2)$ and that the obstacles must be separated from the source by at least a distance equal to 0.3. Then, the "*_avoid" arguments can be used and the resulting function call is

```
[O, a] = CreateRandomDisks(-10, 10, -10, 10, 7, 0.1, 0.5, 0.1, -1, [2;2], 0.3);
```

The disk centered at $(2, 2)$ with radius 0.3 is then avoided. A second option is to set a_void to zero and set the minimal distance dmin_avoid to 0.3

```
[O, a] = CreateRandomDisks(-10, 10, -10, 10, 7, 0.1, 0.5, 0.1, -1, [2;2], 0, 0.3);
```

Remark 3.1. *To check if a disk is correctly placed,* CreateRandomDisks *calls the* CheckPlacement *function which can also be useful for a user who is placing the obstacles manually.*

Removing disks

The function RemoveDisk aims to remove some disks of the geometrical configuration, either disk by disk, by row, by column or by radius. This can be useful for example to delete a row of disks. Here is the syntax

```
[O,a] = RemoveDisk(O_old, a_old, ...);
```

where `O_old` and `a_old` are the centers and radii of the current geometry. Without optional argument, the function has no effect. The available arguments are

- `[O,a] = RemoveDisk(..., 'X', [X1, X2, .., XN]);`
  Remove all the disks with a center which has an abscissa equal to `X1`, `X2`, ..., or `XN`.
- `[O,a] = RemoveDisk(..., 'Y', [Y1, Y2, ..., YN]);`
  Remove all the disks with a center which has an ordinate equal to `Y1`, `Y2`, ..., or `YN`.
- `[O,a] = RemoveDisk(..., 'XY', [[X1;Y1], [X2;Y2], ..., [XN;YN]]);`
  Remove all the disks with a center with coordinates `[X1;Y1]`, `[X2;Y2]`, ..., or `[XN;YN]`.
- `[O,a] = RemoveDisk(..., 'Radius', [a1, a2, ..., aN]);`
  Remove all the disks with a radius equal to `a1`, `a2`, ..., or `aN`.
- `[O,a] = RemoveDisk(..., 'Verbosity', VERBOSITY);`
  set `VERBOSITY` to 0 to avoid display message, to 1 to only show results, and to > 1 to see everything (default).
- `[O,a] = RemoveDisk(..., 'Tol', TOL);`
  Tolerance used for the conditional statement (default 10^{-10}).

Example 1: Remove all the obstacles that are either on the row of with abscissa equal to 1 or on the column with an ordinate equal to 2.5

```
[O,a] = RemoveDisk(O_old, a_old, 'X', 1, 'Y', 2.5);
```

Example 2: Remove the obstacles centered at $(2, 5)$ and $(3, 4)$

```
[O,a] = RemoveDisk(O_old, a_old, 'XY', [2, 3; 5, 4]);
```

Example 3: Create a periodic placement of 11×11 unit disks, separated by a distance equal to 1, then remove the middle line and the middle column, as shown on Figure 3.3. The central disk is moreover centered at $(0, 0)$.

```
bx = 3; by = 3;
Nx = 11; Ny = 11;
O = RectangularLattice(bx, by, Nx, Ny, 'Centered', [0,0]);
a = ones(1, size(O, 2));
[O, a] = RemoveDisk(O, a, 'X', 0, 'Y', 0);
```

3.3.2 Truncation of the Fourier series

To help the user, the formula (2.16) has been coded in μ-diff. The values of N_p are stored in a Matlab vector of size $[1 \times$ `N_scat`$]$ called `M_modes` in μ-diff. Computing `M_modes` only involves the radii of the disks and the wavenumber `k`, which is assumed to be created by the user

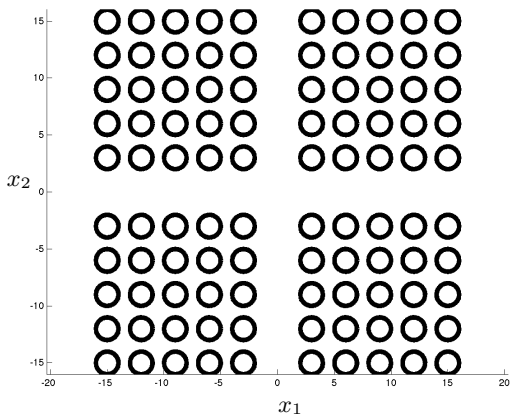


Figure 3.3: Periodic placement with a row and a column deleted.

```
M_modes = FourierTruncation(a, k);
```

The resulting vector is such that `M_modes(p)`$=N_p$, where N_p satisfies (2.16), with obviously a minimal value equal to 0 (which consists in only one mode). If `k` is a vector (which corresponds to one wavenumber per obstacle), then `FourierTruncation` uses formula (2.16) with `k(p)` as the wavenumber. The following options are moreover available

- `M_modes = FourierTruncation(..., 'Min', MIN);`
  To force a minimal value: `M_modes(p)` is then either the min value between `MIN` and formula (2.16).
- `M_modes = FourierTruncation(..., 'Tol', TOL);`
  The tolerance, set by default to 10^{-10}, is then set to `TOL`.

3.3.3 Incident waves

Generalities

Two different incident waves are available in the μ-diff toolbox: the plane wave and the point source wave. The user can build his own incident wave. They are all located in the directory `PreProcessing/IncidentWave/`.
As explained in section 2.1.4, a right-hand side b is decomposed by blocks, each of these blocks representing one obstacle: $b = (b_p)_{p=1,\dots,M}$. A different condition can be applied on two different obstacles (e.g. Dirichlet on Ω_1 and Neumann on Ω_2) or a different integral equation can be considered on each obstacle (e.g. EFIE on Ω_1 and MFIE on Ω_2). To this end, μ-diff builds each block separately thanks to the function `BlockIncidentWave` which computes the vector b_p. According to the input data, the function builds one of the available right-hand sides described in Table 3.1.
On the other hand, the common function `IncidentWave` computes the whole vector b. For all obstacles p, the function `BlockIncidentWave` is called and the whole vector is assembled.
In addition, for all the incident waves, an interface function is available. These easy-to-use interfaces build the whole vector on only one pattern (trace of plane wave, normal derivative of a point source wave, ...). They are located in the **`interface/`** directory and their names allow an easy interpretation (see also table 3.1, column μ-diff name): `PlaneWave`, `PointSource`, `DnPlaneWave`,.... Let us recall that the help informations of the interface functions contains the mathematical description of the incident wave.

Value	μ-diff function name	Param	Description
1	`PlaneWave`	`beta_inc`	trace of a plane wave of angle of direction `beta_inc`. A plane wave is defined by $e^{ik(\cos(\beta)x_1+\sin(\beta)x_2}$
2	`DnPlaneWave`	`beta_inc`	normal derivative of a plane wave of angle of incidence `beta_inc`
3	`PointSource`	`XS`	trace of the wave emitted by a point source placed at `XS`. Such a wave is defined in μ-diff by $\frac{i}{4}H_0^{(1)}(k\|\mathbf{x}-\mathbf{x}_s\|)$
4	`DnPointSource`	`XS`	normal derivative trace of the wave emitted by a point source placed at `XS`
5	`PlaneWavePrecond`	`beta_inc`	same as `PlaneWave` but multiplied by the inverse of the single-layer block diagonal operator (see section 2.1.3 on the single-scattering preconditioner)
6	`DnPlaneWavePrecond`	`beta_inc`	same as `DnPlaneWave` but multiplied by the inverse of the double-layer block diagonal operator

Table 3.1: Right-hand sides already coded in μ-diff.

The two main functions, `BlockIncidentWave` and `IncidentWave`, are now detailed.

`BlockIncidentWave`

This function computes the block vector of **the opposite of** the coefficients of an incident wave, either the trace of the normal derivative trace, on one of the obstacles, in the Fourier bases. Its syntax is

```
Bp = BlockIncidentWave(Op, ap, Np, k, TypeOfWave, Param);
```

where `TypeOfWave` is a scalar value specifying the incident wave (see table 3.1) and `Param` is the parameter of the wave: angle of direction, position of a point source, ... The returned value `Bp` is a column vector of length $2\texttt{Np}+1$.

`IncidentWave`

The function call is the following

```
B = IncidentWave(O, a, M_modes, k, TypeOfWave, Param)
```

The resulting vector `B` is of size $\sum_{p=1}^{M}(2\texttt{M\_modes}(p)+1)$. The value `Param` is the same as for `BlockIncidentWave` whereas `TypeOfWave` is a vector of size M, where `TypeOfWave`(p) is the fixed choice for the block b_p. In other word, the block b_p is built by calling the function with the arguments `BlockIncidentWave(Op, ap, Np, k, TypeOfWave, Param);`. To simplify, if `TypeOfWave` is a scalar value, then it is considered as a vector with the same scalar value.

Example 1: Building a vector associated to the trace of an incident plane wave of direction `beta_inc` is done thanks to the following command (the "1 argument" refers to as `PlaneWave`)

```
B = IncidentWave(O, a, M_modes, k,  1, beta_inc);
```

or by using the interface function

```
B = PlaneWave(O, a, M_modes, k, beta_inc);
```

Example 2: For two obstacles and a point source centered at $(1, 2)$, if b_1 is the trace of the wave and b_2 is the normal derivative trace, then

```
B = IncidentWave(O, a, M_modes, k,  [3;4],[1;2]);
```

In other words, this builds the vector (`[3,4]` can be translated as `PointSource, DnPointSource`)

$$b = \begin{pmatrix} -u^{\text{inc}}|_{\Gamma_1} \\ -\partial_{\mathbf{n}} u^{\text{inc}}|_{\Gamma_2} \end{pmatrix},$$

where

$$u^{\text{inc}} = \frac{i}{4} H_0^{(1)}(k\|\mathbf{x} - \mathbf{x}_s\|),$$

with $\mathbf{x}_s = [1, 2]$.

Remark 3.2. *Remember that the resulting vector corresponds to the opposite of the trace or normal derivative trace!*

3.4 Integral operators

3.4.1 Generalities

The functions defining the integral operators are available in the directory `IntOperators/` which has the `Dense/` and `Sparse/` subdirectories for the dense (matrix) and sparse (@function) representations of the four basic integral operators used in scattering, i.e. $\mathbb{L}$, $\mathbb{M}$, $\mathbb{N}$ and $\mathbb{D}$, given in their infinite dimensional operator versions by respectively (2.2), (2.3), (2.4) and (2.5). Preconditioned versions of the operators by their single-scattering operators [24] are also defined. Following Proposition 1.9, only $\widehat{L}^{-1}L$ (EFIE Dirichlet) and $\widehat{D}^{-1}D$ (EFIE Neumann) are provided since they are the only ones needed for the Dirichlet and Neumann problems.
Two different types of storage can be used within the μ-diff toolbox: dense and sparse. The assembly of the matrix is almost the same in both cases. The way μ-diff is developed is close to the mathematics for the assembly process, in the sense that the matrix $\mathbb{A}$ of an integral operator is built block-by-block ($\mathbb{A}^{p,q}$). For both storages, two main functions exist: a "block" function (`BlockIntegralOperator` and `SpBlockIntegralOperator`) and a global matrix function (`IntegralOperator` and `SpIntegralOperator`) which assemble all the blocks into a matrix. This separation allows the user to either build a "simple" matrix for one operator or to construct a more "complex" matrix where each block represents a different operator or a linear combination of them. Let us also note that all the operators have interface functions (located in `Dense/Interface` or `Sparse/Interface` folders). For example, `SingleLayer` is the interface function of `IntegralOperator` to build the single-layer operator, and `BlockSingleLayer` the one for `BlockIntegralOperator`. The same applies for the sparse storage with a prefix `Sp`: `SpSingleLayer`, `SpBlockSingleLayer`.

Int. Op.	Identifier	Function	Definition for $\mathbf{x} \in \Gamma$
-	0	-	zero operator (null matrix)
I	1	`Identity`	identity
L	2	`SingleLayer`	$L\rho(\mathbf{x}) = \int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\mathbf{y}$
M	3	`DoubleLayer`	$M\lambda(\mathbf{x}) = -\int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\mathrm{d}\mathbf{y}$
N	4	`DnSingleLayer`	$N\rho(\mathbf{x}) = \partial_{\mathbf{n_x}} \int_\Gamma G(\mathbf{x},\mathbf{y})\rho(\mathbf{y})\mathrm{d}\mathbf{y}$
D	5	`DnDoubleLayer`	$D\lambda(\mathbf{x}) = -\partial_{\mathbf{n_x}} \int_\Gamma \partial_{\mathbf{n_y}} G(\mathbf{x},\mathbf{y})\lambda(\mathbf{y})\mathrm{d}\mathbf{y}$
$\hat{L}^{-1}L$	6	`PrecondDirichlet`	single-layer preconditioned by its diagonal
$\hat{D}^{-1}D$	7	`PrecondNeumann`	double-layer preconditioned by its diagonal

Table 3.2: Available integral operators in μ-diff, their (unique) identifier, function name (interface) and the mathematical definition. The zero operator function does not have an interface function and the sparse version is obtained by adding the prefix `Sp` (`SpSingleLayer`, `SpDnDoubleLayer`,...). The block interface functions are also prefixed by "`Block`" (`SpBlockSingleLayer`, `BlockSingleLayer`,...).

The dense storage should be preferred for solving small scale problems (most particularly low frequency problems), when the memory storage and the CPU cost is not a problem, where the sparse storage must be used when the limits are reached. For the sparse storage, which is detailed later, the block matrices are stored through vectors and the matrix-vector product is then fast when an iterative Krylov subspace solver is used. This is essentially a suitable strategy when a sufficiently high number of modes N_p is required per obstacle, i.e. for large enough wave numbers k. The drawback is that some instabilities may arise in the numerical process if the truncations of the Fourier series are not done correctly. The formula (2.16) provides a stable result following [2].

3.4.2 Available integral operators and numbering

As for the incident wave, for each operator, there exists a function that builds the whole operator for the multiple scattering problem, both for the dense and sparse storages. The available operators are listed in Table 3.2, with their unique identifier (integer), their associated name in μ-diff (useful to get their interface functions) and their definitions.

3.4.3 Dense storage

Building a block $\mathbb{A}^{p,q}$

An elementary block matrix in a global matrix can be created by using the following function `BlockIntegralOperator`

```
Apq = BlockIntegralOperator(Op, ap, Np, Oq, aq, Nq, k, TypeOfOperator, Weight);
```

The `Weight` argument is optional and set to 1 by default. The quantity `TypeOfOperator` specifies the integral operator to compute, thanks to the numbering of Table 3.2. If `TypeOfOperator`

is a scalar (e.g. $= 2$), then the resulting matrix `Apq` is the elementary matrix of the associated operator (e.g. $\mathbb{L}^{p,q}$). If `TypeOfOperator` is a row (e.g. `[1,3]`), the sum of the two operators is computed (e.g. $\mathbb{I}^{p,q}+\mathbb{M}^{p,q}$). Finally, the `Weight` quantity (of the same size as `TypeOfOperator`) is the constant that is used to multiply the block and hence

$$\mathbb{A}^{p,q} = \sum_{\ell=1}^{N} \texttt{Weight}(\ell).\texttt{Operator}(\ell)$$

where $\texttt{Operator}(\ell)$ is one of the integral operators.

Example 1: Build the single-layer block $\mathbb{L}^{p,q}$

```
Apq = BlockIntegralOperator(Op, ap, Np, Oq, aq, Nq, k, 1);
```

Example 2: Build the whole matrix $0.5 \times \mathbb{I}^{p,q} + \mathbb{N}^{p,q}$, appearing in the MFIE (1.16)

```
Apq = BlockIntegralOperator(Op, ap, Np, Oq, aq, Nq, k, [1, 4], [0.5, 1]);
```

Example 3: Compute the sum of the four blocks: $0.5\times\mathbb{L}^{p,q}+1.5\times\mathbb{M}^{p,q}+2.5\times\mathbb{N}^{p,q}+3.5\times\mathbb{D}^{p,q}$

```
Apq = BlockIntegralOperator(Op, ap, Np, Oq, aq, Nq, k, [2, 3, 4, 5],
                            [0.5, 1.5, 2.5, 3.5]);
```

Assembling the matrix $\mathbb{A}$

Now that the construction of an elementary block matrix is well-understood, building a global matrix is easy thanks to the common function

```
A = IntegralOperator(O, a, M_modes, k, TypeOfOperator, Weight);
```

As for `BlockIntegralOperator`, the quantity `Weight` is optional and set to 1 by default. Roughly speaking, `IntegralOperator` creates a loop on all the obstacles p and q, launches the following command

```
Apq = BlockIntegralOperator(O(:,p), a(p), M_modes(p), O(:,q), a(q), M_modes(q),
                                                      k, Tpq, Wpq);
```

and places `Apq` in the expected elementary block matrix of the global matrix. The quantities `Tpq` and `Wpq` are given by `TypeOfOperator` and `Weight` such that (`Weight` is of the same size as `TypeOfOperator` and so `Wpq` follows the same rules as `Tpq`)

- If `TypeOfOperator` is a scalar then `Tpq` = `TypeOfOperator`.
- If `TypeOfOperator` is a row or a column vector then `Tpq` is an array given by `Tpq` = `TypeOfOperator`.
- If `TypeOfOperator` is a matrix then `Tpq` = `TypeOfOperator(p,q)`.
- If `TypeOfOperator` is a three-dimensional array, then `Tpq` is an array given by `Tpq(:)` = `TypeOfOperator(p,q,:)`.

Example 1: The single-layer potential L is

```
L = IntegralOperator(O, a, M_modes, k, 2);
```

Example 2: The MFIE operator $0.5 \times I + N$ for a Dirichlet boundary condition (1.16) is

```
A_MFIE = IntegralOperator(O, a, M_modes, k, [1,4], [0.5,1]);
```

Example 3: The following two-obstacles operator is

$$\begin{pmatrix} 0.5 \times I_{1,1} + M_{1,1} & L_{1,2} \\ D_{2,1} & 0.5 \times I_{2,2} + N_{2,2} \end{pmatrix}$$

can be computed by using a three-dimensional array and the null operator

```
TypeOfOp = zeros(2,2,2); Weight = zeros(2,2,2);
TypeOfOp(:, 1,1) = [1, 3]; Weight(:, 1,1) = [0.5, 1]; %block A_{1,1}
TypeOfOp(:, 1,2) = [2, 0]; Weight(:, 1,2) = [1, 0]; %block A_{1,2}
TypeOfOp(:, 2,1) = [5, 0]; Weight(:, 2,1) = [1, 0]; %block A_{2,1}
TypeOfOp(:, 2,2) = [1, 4]; Weight(:, 2,2) = [0.5, 1]; %block A_{2,2}
A = IntegralOperator(O, a, M_modes, k, TypeOfOp, Weight);
```

Example 4: The Brakhage-Werner Integral Equation (BWIE) for a Dirichlet problem (1.21) is solved by writing

```
k = 1;
beta_inc = pi;
eta = i/k;
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
A = IntegralOperator(O, a, M_modes, k, [1, 2, 3], [0.5, -eta_BW, -1]);
psi = A \ Uinc;
```

3.4.4 Sparse storage

Storing the matrices in a sparse way leads to a significant reduction in memory storage and the ability to get access to fast matrix-vector product evaluations. The linear system must then be solved by an iterative solver since the global matrix is no longer built. Combining linearly the matrices is still possible but this is realized during the computation of the matrix-vector products. Indeed, summing two matrices for two different integral operators is not guaranteed to keep the particular matrix structure. If the problem involves (at least) two different integral operators, they must hence be computed separately.

Compressed storage

To explain how the matrix is stored by using the μ-diff sparse storage, let us consider that $\mathbb{A}$ is a matrix corresponding to one of the four integral operators L, M, N or D. The matrix $\mathbb{A}$ has the following special structure (for $p, q = 1, \ldots, M$ and $p \neq q$)

- $\mathbb{A}^{p,p}$ is diagonal,
- $\mathbb{A}^{p,q}$ is full and can be written as $\mathbb{A}^{p,q} = \mathbb{A}_L^{p,q}\mathbb{T}^{p,q}\mathbb{A}_R^{p,q}$, where $\mathbb{A}_L^{p,q}$ and $\mathbb{A}_R^{p,q}$ are diagonal and called respectively the left and right parts, and $\mathbb{T}^{p,q} = (\mathbb{T}_{m,n}^{p,q}))$, with

$$\mathbb{T}_{m,n}^{p,q} = i\pi e^{i(n-m)\alpha_{pq}} H_{n-m}^{(1)}(kb_{pq}),$$

  is a Toeplitz matrix: $\mathbb{T}_{m,n}^{p,q} = \mathbb{T}_{m+1,n+1}^{p,q}$.

The idea is that the diagonal matrices can be stored as vectors containing the diagonal elements. A matrix-vector product between a diagonal matrix and a vector is then simply an element-by-element multiplication. The Toeplitz matrix can also be stored in a compressed form as a vector and the matrix-vector product is handled by a cross-correlation based on the Fast Fourier Transform (FFT).
The diagonals parts are stored in the three-dimensional arrays $\mathbb{A}_L$ and $\mathbb{A}_R$ ("L" for Left part and "R" for Right part). The left part $\mathbb{A}_L$ also includes the diagonal part of the matrix

$$\mathbb{A}_L(:,p,q) = \begin{cases} \text{diag}(\mathbb{A}^{p,p}) & \text{if } p = q, \\ \text{diag}(\mathbb{A}_L^{p,q}) & \text{otherwise,} \end{cases} \qquad \mathbb{A}_R(:,p,q) = \begin{cases} 0 & \text{if } p = q, \\ \text{diag}(\mathbb{A}_R^{p,q}) & \text{otherwise.} \end{cases}$$

The matrix $\mathbb{T}^{p,q}$ of size $(2N_p+1)\times(2N_q+1)$ is then also compressed as a Toeplitz matrix following

$$\mathbb{T}^{p,q} = \begin{pmatrix} t_1 & t_2 & t_3 & \cdots & t_{(2N_q+1)} \\ t_{(2N_q+1)+1} & t_1 & t_2 & \cdots & t_{(2N_q+1)-1} \\ t_{(2N_q+1)+2} & t_{(2N_q+1)+1} & t_1 & \cdots & t_{(2N_q+1)-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{(2N_q+1)+(2N_p+1)-1} & t_{(2N_q+1)+(2N_p+1)-2} & t_{(2N_q+1)+(2N_p+1)-3} & \cdots & t_{(2N_q+1)-(2N_p+1)+1} \end{pmatrix}$$

Clearly, the root vector containing the first row and the first column is enough to rebuild the matrix. This root vector, called $\mathbb{A}_M^{p,q}$, is of size $(2N_p+1)+(2N_q+1)-1 = 2N_p+2N_q+1$ and such that

$$\mathbb{A}_M^{p,q} = \begin{pmatrix} t_{(2N_q+1)+(2N_p+1)-1} \\ t_{(2N_q+1)+(2N_p+1)-2} \\ \vdots \\ t_{(2N_q+1)+2} \\ t_{(2N_q+1)+1} \\ t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_{(2N_q+1),1} \end{pmatrix}.$$

Finally, all these root vectors are stored in the three-dimensional array $\mathbb{A}_M$

$$\mathbb{A}_M(:,p,q) = \begin{cases} 0 & \text{if } p = q, \\ \mathbb{A}_M^{p,q} & \text{otherwise.} \end{cases}$$

The global matrix $\mathbb{A}$ is thus stored through three different parts: $\mathbb{A}_L$, $\mathbb{A}_M$ and $\mathbb{A}_R$. From the computer point of view, as a three-dimensional array has a fixed size in each direction, the second and third dimensions of the arrays are both of size `N_scat` and the length in the first dimension is given by

3D-array	Length in the dimension...		
	1	2	3
$\mathbb{A}_L$	$\max_p(2N_p+1)$	N_scat	N_scat
$\mathbb{A}_M$	$\max_p \max_{q\neq p}[(2N_p+1)+(2N_q+1)-1]$	N_scat	N_scat
$\mathbb{A}_R$	$\max_p(2N_p+1)$	N_scat	N_scat

With this constraint, the vector $\mathbb{A}_L(:,p,q)$ can be larger than $2N_p+1$ and thus $\mathbb{A}_L^{p,q}$ must be extracted: $\mathbb{A}_L^{p,q} = \mathbb{A}_L(1:2N_p+1,p,p)$. The same occurs for $\mathbb{A}_M$ and $\mathbb{A}_R$.
Finally, these three-dimensional arrays are merged into a cell A, representing the matrix $\mathbb{A}$, such that, by using the Matlab notations

$$\texttt{A}\{1\} = \mathbb{A}_L, \qquad \texttt{A}\{2\} = \mathbb{A}_M, \qquad \texttt{A}\{3\} = \mathbb{A}_R.$$

Fast matrix-vector products

The matrix-vector product between $\mathbb{A}$ and a vector $\mathbf{X}$ is divided into different elementary operations. Let us consider $\mathbf{Y} = \mathbb{A}\mathbf{X}$ and more particularly the p^{th} component

$$\mathbf{Y}_p = \sum_q \mathbb{A}^{p,q}\mathbf{X}_q.$$

By using the previous notations, we have

$$\mathbf{Y}_p = \mathbb{A}_L^{p,p}\mathbf{X}_p + \sum_{q\neq p} \mathbb{A}_L^{p,q}(\mathbb{A}_M^{p,q}(\mathbb{A}_R^{p,q}\mathbf{X}_q)).$$

Since $\mathbb{A}_L^{p,p}$, $\mathbb{A}_L^{p,q}$ and $\mathbb{A}_R^{p,q}$ are diagonal matrices (that are stored as vectors), the corresponding matrix-vector products are easy to compute. The only difficulty concerns $\mathbb{A}_M^{p,q}\mathbf{Z}_q$ (where $\mathbf{Z}_q = \mathbb{A}_R^{p,q}\mathbf{X}_q$). This can however be achieved efficiently. Indeed, the discrete cross-correlation product (xcorr) between $\mathbb{A}_M^{p,q}$ and $\mathbf{Z}_q$ gives

$$\widetilde{\mathbf{W}}_q = \texttt{xcorr}(\mathbf{Z}_q, \overline{\mathbb{A}_M^{p,q}}),$$

where the bar denotes the complex conjugate of a complex number. The result $\mathbf{W}_q$ is then extracted from $\widetilde{\mathbf{W}}_q$ by

$$\mathbf{W}_q = \widetilde{\mathbf{W}}_q(2N_q+1 : 2N_q+1+2N_p).$$

The matrix-vector product between $\mathbb{A}$ and $\mathbf{X}$ is then done in a fast way (cross-correlation is efficiently evaluated through the FFT).

Assembling the matrix

The assembly process is very similar to the dense one, the difference being the prefix Sp. The block function is then called by

```
SpBlockIntegralOperator(Op, ap, Np, Oq, aq, Nq, Nmax, k, TypeOfOperator, Weight);
```

The function returns three vectors corresponding to $\mathbb{A}_L^{p,q}$, $\mathbb{A}_M^{p,q}$ and $\mathbb{A}_R^{p,q}$. The major difference here is that **no linear combination of operators can be done during the assembly process** for the sparse representation. The quantity TypeOfOperator cannot hence be a vector but must be a scalar. The assembly process of the global matrix can be done through

```
A = SpIntegralOperator(O, a, M_modes, k, TypeOfOperator, Weight);
```

The result is a Matlab `cell` of three components, where each component is a three-dimensional array. The quantity `TypeOfOperator` can be a scalar or a matrix (not a three-dimensional array nor a vector!). If `TypeOfOperator` is a matrix, then the block $\mathbb{A}^{p,q}$ is assumed to be of type `TypeOfOperator(p,q)`. The linear combination of operators can still be done, but must be specified in the matrix-vector product (see below). Note that a function exists to add the (weighted-)identity to a sparse operator (see below).

Example 1: Creation of μ-diff sparse matrix of the single-layer operator L

```
L = SpIntegralOperator(O, a, M_modes, k, 2);
```

`L` is now a cell with three components! Note the interface function can also be used

```
L = SpSingleLayer(O, a, M_modes, k);
```

Example 2: For two obstacles, building the following matrix

$$\begin{pmatrix} M_{1,1} & 0.5 \times L_{1,2} \\ 1.5 \times D_{2,1} & 2.5 \times N_{2,2} \end{pmatrix}$$

can be done with the commands

```
TypeOfOp = zeros(2,2); Weight = zeros(2,2);
TypeOfOp(1,1) = [3]; Weight(1,1) = [1]; %block A_{1,1}
TypeOfOp(1,2) = [2]; Weight(1,2) = [0.5]; %block A_{1,2}
TypeOfOp(2,1) = [5]; Weight(2,1) = [1.5]; %block A_{2,1}
TypeOfOp(2,2) = [4]; Weight(2,2) = [2.5]; %block A_{2,2}
A = SpIntegralOperator(O, a, M_modes, k, TypeOfOp, Weight);
```

Assembling: adding the identity

It is possible to add the identity (multiplied by a constant) to a sparse operator

```
A = SpAddIdentity(A, alpha, M_modes);
```

which simply returns $A = A + \alpha I$.

Example 1: If `L` is the sparse representation of the single-layer operator, then

```
L = SpIntegralOperator(O, a, M_modes, k, 2);
alpha_L = SpAddIdentity(L, 0.5, M_modes);
```

computes $0.5 \times L$.

Sparse matrix-vector product

A matrix-vector product $\mathbf{Y} = \mathbb{A}\mathbf{X}$ is done thanks to

```
Y = SpMatVec(X, M_modes, ListOfOperators, Weight);
```

where `Weight` is optional. `ListOfOperators` is a **Matlab cell** (not an array! Make use of $\{\cdot\}$ instead of $[\cdot]$) containing all the sparse matrices that the user wants to involve in the computational process. The linear combination of the operators is done at that time for each matrix-vector product.

Example 1: Computing $\mathbf{Y} = \mathbb{L}\mathbf{X}$ is done as follows

```
SpL = SpIntegralOperator(O, a, M_modes, k, 2);
Y = SpMatVec(X, M_modes, SpL);
```

Example 2: Calculating $\mathbf{Y} = (0.5 \times I + N)\mathbf{X}$, where I is the identity (**Beware the** $\{\cdot\}$ **!**), consists in the following sequence of function calls

```
SpI = SpIdentity(O, a, M_modes);
SpN = SpDnSingleLayer(O, a, M_modes, k);
Y = SpMatVec(X, M_modes, {SpI, SpN}, [0.5, 1]);
```

This can also be done thanks to the `SpAddIdentity` function as

```
SpN = SpDnSingleLayer(O, a, M_modes, k);
A = SpAddIdentity(SpN, 0.5, M_modes);
Y = SpMatVec(X, M_modes, A);
```

Solving a linear system

Now that the matrices and the right-hand side have been built, the sparse linear system can be solved iteratively. If one uses for example the GMRES solver [22], the syntax is then...

Example 1: ...for the Dirichlet EFIE (1.15) : $L\rho = -u^{\text{inc}}|_\Gamma$

```
SpL = SpSingleLayer(O, a, M_modes, k);
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
rho = gmres(@(X)SpMatVec(X, M_modes, SpL), Uinc);
```

Example 2: ...for the Dirichlet MFIE (1.16) : $(I/2 + N)\rho = -\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma$

```
SpN = SpDnSingleLayer(O, a, M_modes, k);
SpI = SpIdentity(O, a, M_modes);
DnUinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
rho = gmres(@(X)SpMatVec(X, M_modes, {SpI, SpN}, [0.5, 1]), DnUinc);
```

or by using the `SpAddIdentity` function

```
SpAMFIE = SpDnSingleLayer(O, a, M_modes, k);
SpAMFIE = SpAddIdentity(SpAMFIE, 0.5, M_modes);
DnUinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
rho = gmres(@(X)SpMatVec(X, M_modes, SpAMFIE), DnUinc);
```

Example 3: ...for the Dirichlet BWIE (1.21) : The operators L and M must be computed separately and $0.5I$ can be added to $-M$ directly:

```
k = 1;
beta_inc = pi;
eta = i/k;
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
SpL = SpSingleLayer(O, a, M_modes, k);
SpM = - SpDoubleLayer(O, a, M_modes, k);
SpIminusM = SpAddIdentity(SpM,0.5);
psi = gmres(@(X)SpMatVec(X, M_modes, {SpL, SpIminusM}, [-eta_BW, 1]), Uinc);
```

3.5 Post-Processing

Now that the system has been solved, the next step is to display some results. The μ-diff toolbox proposes some post-processing features such as computing the far-field (fast) or the near-field of the wave on a grid (Matlab `meshgrid`) or only at some points. In addition, other possibilities are offered such as drawing the disks on a figure or the incident field. Both the near- and far-field computations can be done for a linear combination of a single- and double-layer potentials $\mathscr{L}\rho + \mathscr{M}\lambda$, only one of them (*e.g.* $\mathscr{L}\rho$) or with the same density $(\mathscr{L} + \mathscr{M})\psi$, with a single command line. All post-processing functions are located in the `PostProcessing/` directory.

Far-field

The far-field of the combination of the single- and double-layer potentials

$$\sum_{p=1}^{M} \eta_p \mathscr{L}_p \rho_p + \gamma_p \mathscr{M}_p \lambda_p$$

is given by the equation (2.12). The corresponding μ-diff function is `FarField`

```
F = FarField(O, a, M_modes, k, theta, Density, Weight);
```

where the parameters are

- `theta` is the vector of receiving angles (in radians)
- `Density` is either ρ, λ or both. If `Density` is a column vector then `Density(:,2)` = `Density(:,1)` and only one density is used both for the single- and the double-layer potentials contribution. If `Density` has 2 columns, then `Density(:,1)` is considered to be the single-layer density and `Density(:,2)` the double-layer density.
- `Weight` is the weight vector to apply to the M volume integral operators (η_p and γ_p). The quantity `Weight` is of size either $[1 \times 2]$ or $[1 \times M]$. The first column of `Weight` is applied to the single-layer potentials and the second to the double-layer potentials: `Weight(p,1)`$=\eta_p$

and `Weight(p,2)`$=\gamma_p$. If `Weight` is a row of length 2, then `Weight(1)`$=\eta_p$ and `Weight(2)`$=\gamma_p$ for all p (the coefficient are the same for all obstacles).

With these notations, the resulting far-field is

$$\sum_{p=1}^{\texttt{N\_scat}} \texttt{Weight(p, 1)}\mathscr{L}\texttt{Density(:,1)} + \texttt{Weight(p,2)}\mathscr{M}\texttt{Density(:,2)}.$$

Example 1: the Dirichlet EFIE (1.15): the scattered field u reads as $u = \mathscr{L}\rho$ and the far-field is then computed by

```
F = FarField(O, a, M_modes, k, theta, rho, [1,0]);
```

Example 2: the Dirichlet BWIE (1.21): the scattered field u is given by $u = (-\eta\mathscr{L} - \mathscr{M})\psi$ and the corresponding far-field by

```
eta = i/k;
F = FarField(O, a, M_modes, k, theta, psi, [-eta,-1]);
```

Example 3: if the densities ρ and λ are known and if the user needs to compute for example

$$\sum_{p=1}^{M} p\mathscr{L}_p\rho_p - p\mathscr{M}_p\lambda_p,$$

then this can be done by the function call

```
F = FarField(O, a, M_modes, k, theta, [rho, lambda], [[1:p].', -[1:p].']);
```

Remark 3.3. *Let us remark that the `FarField` function is also interfaced with the ready-to-use function: `FarFieldSingleLayer` and `FarFieldDoubleLayer`. These two functions are located in the `PostProcessing/FarField/Interface` directory.*

Radar Cross Section (RCS)

The Radar Cross Section (RCS) can be computed either from the far-field with `FarField_to_RCS` or directly from the density with `RCS`. In μ-diff, the RCS σ is obtained by (F being the far-field)

$$\sigma = 10\log_{10}(2\pi|F|^2).$$

The computation is realized by the function call

```
R = FarField_to_RCS(F);
```

where `F` has been computed by the `FarField` function. Otherwise, the RCS can be computed directly from the densities by

```
R = RCS(O, a, M_modes, k, theta, Density, Weight);
```

where the arguments are exactly the same as for the far-field. In fact, `RCS` first calls `FarField` and then `FarField_to_RCS`.

Remark 3.4. *The `RCS` function is also interfaced with the two following ready-to-use functions `RCSSingleLayer` and `RCSDoubleLayer`. Both of them are located in the directory called `PostProcessing/FarField/Interface`.*

Near-field: inside and outside the obstacles

The near-field of a potential is given by equations (2.8) and (2.9) outside the obstacles and by (2.10) and (2.11) inside. In the same way, μ-diff separates the outside and the inside computations.
Let us first explain how to compute the field outside the obstacles. The user only needs to create a mesh by using e.g. `meshgrid` and launch `ExternalPotential` through the syntax

```
U = ExternalPotential(X, Y, O, a, M_modes, k, Density, Weight, OPTIONS);
```

The resulting matrix `U` has zero value inside the obstacle. By default, the computation is not realized on the boundary of the obstacles (this can be changed thanks to the options). The arguments are

- `Density` and `Weight` are exactly the same as for `FarField`,
- the quantities `X` and `Y` are the grid points which are created by the Matlab `meshgrid` function,
- the `OPTIONS` must be chosen from the options of the potential, see below (after the interior potential).

Let us consider now the computation inside the obstacle Ω_p. For the interior field, the only contribution is assumed to be of the form

$$\mathscr{L}_p\rho_p + \mathscr{M}_p\lambda_p.$$

In other words, the contributions of the other obstacles are not taken into account. In the same way as for the external potential, the computation is realized in μ-diff by the function call

```
U = InternalPotential(X, Y, O, a, M_modes, k, Density, Weight, OPTIONS);
```

where `U` has zero value outside the obstacles and by default on the boundaries too. The arguments are exactly the same as `ExternalPotential`.
Finally, the common options for `ExternalPotential` and `InternalPotential` are described below. For the sake of clarity, the examples are shown with only `ExternalPotential` even if they also apply for `InternalPotential`

- `ExternalPotential(..., 'Verbosity', VERBOSITY)`
  `VERBOSITY` is a scalar, which when set to 0, stop displaying message (default = 1)
- `ExternalPotential(..., 'OnBoundary', ONBOUNDARY)`
  if `ONBOUNDARY` is set to 1 then the boundary is also covered (beware of the jump relation...) (default = 0)

Example 1: compute the single-layer potential $u = \mathscr{L}\rho$ on a grid

```
XX = [-10:0.1:10];
YY = [-10:0.1:10];
[X,Y] = meshgrid(XX,YY);
U = ExternalPotential(X, Y, O, a M_modes, k, rho, [1,0]);
```

Example 2: consider two obstacles and compute

$$\mathscr{L}_1\rho_1 + 0.5 \times \mathscr{L}_2\rho_2 + 1.5 \times \mathscr{M}_1\lambda_1 - 1.5 \times \mathscr{M}_2\lambda_2$$

inside the obstacle, where the densities have already been computed

```
XX = [-10:0.1:10];
YY = [-10:0.1:10];
[X,Y] = meshgrid(XX,YY);
U = InternalPotential(X, Y, O, a M_modes, k, [rho, lambda], [1, 1.5 ; 0.5, -1.5]);
```

Remark 3.5. *The near-field functions* `ExternalPotential` *and* `InternalPotential` *are also interfaced by:* `ExternalSingleLayerPotential`, `InternalSingleLayerPotential`, *for the single-layer and by* `ExternalDoubleLayerPotential` *and* `InternalDoubleLayerPotential` *for the double-layer potential. These four ready-to-use functions are located in the following folder* ***`PostProcessing/NearField/Interface`****.*

Incident wave

The incident waves can also be computed on a grid by using

```
Uinc = IncidentWaveOnGrid(X, Y, k, TypeOfWave, Param);
```

where, like for the `IncidentWave` function from the pre-processing part (see §3.3.3), we have

- `X` and `Y` are two matrices coming from the `meshgrid` Matlab function,
- `Param` is either the incidence angle of a (scalar) plane wave or the location of a point source ($[2 \times 1]$ vector),
- `TypeOfWave` is either a `char` or a scalar value.

Example: computation of a plane wave with an incidence angle equal to `beta_inc` $= \pi$

```
beta_inc = pi;
XX = [-10:0.1:10];
YY = [-10:0.1:10];
[X,Y] = meshgrid(XX,YY);
Uinc = IncidentWaveOnGrid(X, Y, k, 'PlaneWave', beta_inc);
```

The last line could also be (switch `'PlaneWave'` with its identifier 1)

```
Uinc = IncidentWaveOnGrid(X, Y, k, 1, beta_inc);
```

Geometry: drawing the obstacles and creating mask matrices

The μ-diff toolbox also proposes some functions for the post-processing of the obstacles. Drawing the circular cylinders can be done by calling the following function

```
PlotCircles(O, a, fig_index, OPTIONS);
```

where `fig_index` is the Figure handle and the `OPTIONS` are

- `PlotCirclesOnFigure(..., 'Color', COLOR)`: apply the color `COLOR` to lines (same as the plot function)
- `PlotCirclesOnFigure(..., 'LineWidth', LINEWIDTH)`: set the line width to `LINEWIDTH` (same as the plot function)
- `PlotCirclesOnFigure(..., 'zdata', ZDATA)`: set the zdata of the figure to `ZDATA` (same as the plot function)

When drawing disks on a figure containing some values, do not forget to set the zdata to the max value! An example is given below

```
figure(1);
surf(X,Y,U);
PlotCirclesOnFigure(O, a, 1, 'zdata', max(max(U)));
```

If X and Y are obtained by `meshgrid`, let us define the mask matrix S by

$$S(i,j) = \begin{cases} 0 & \text{if } (X(i,j), Y(i,j)) \text{ is outside the obstacles,} \\ p & \text{if } (X(i,j), Y(i,j)) \in \Omega_p, \\ p + 0.5 & \text{if } (X(i,j), Y(i,j)) \in \Gamma_p. \end{cases}$$

This class of matrices is handy when plotting the potentials and is actually used by the two functions `ExternalPotential` and `InternalPotential`. The function `MaskMatrixObstacles` which provides the matrix M has the following syntax

```
S = MaskMatrixObstacles(X, Y, O, a);
```

Let us explain how to extract the boundary points. In the same way as for the `MaskMatrixObstacles` function, `BoundaryOfObstacles` leads to a matrix G where

$$G(i,j) = \begin{cases} 0 & \text{if } (X(i,j), Y(i,j)) \text{ is outside the obstacles,} \\ 0 & \text{if } (X(i,j), Y(i,j)) \in \Omega_p, \\ p & \text{if } (X(i,j), Y(i,j)) \in \Gamma_p. \end{cases}$$

Similarly, G is computed by

```
G = BoundaryOfObstacles(X, Y, O, a);
```

3.6 Examples available in μ-diff

Some examples are available in the `Examples/` directory and especially in `Examples/Benchmark`, where each file is independent and solves a different problem (Dirichlet, Neumann or penetrable). They should also be launched to check that μ-diff is correctly installed on your computer and provides the expected results. In the next chapter, we present some standard examples of μ-diff scripts.

Chapter 4

Simple examples of multiple scattering problems solved with μ-diff

Contents

The aim of this chapter is to provide some examples of multiple scattering problems solved by the μ-diff toolbox. The impenetrable case with a Dirichlet, a Neumann or a mixed of both boundary conditions set on the boundaries of the obstacles are fully treated. For penetrable obstacles, an example of the implementation of equation (1.36) is provided.

4.1 The Dirichlet boundary-value problem

Let us consider the scattering problem by a collection of sound-soft obstacles

$$\begin{cases} (\Delta + k^2)u &= 0, & \text{in } \Omega^+, \\ u &= -u^{\text{inc}}, & \text{on } \Gamma, \\ & u \text{ outgoing}, \end{cases}$$

with $\Omega^- = \bigcup_{p=1}^{M} \Omega_p^-$. We propose to solve this problem through various integral equations: the EFIE (1.15), the MFIE (1.16), the CFIE (1.19) and the single-scattering preconditioned integral equations (1.32). We show how to use both the full and sparse storages of the matrices. Before starting, we recommend to use the single-scattering preconditioned integral equation as presented in §4.1.5. Indeed, the resulting system is well-posed and is well-conditioned leading to an efficient solution by a Krylov subspace iterative solver.

4.1.1 Pre-processing

Let us first consider a collection of three sound-soft unit circular cylinders. The wavenumber is $k = 2\pi$ and the direction of incidence of the wave is $\beta = 0$ degree. The resulting μ-diff pre-processing code for setting these parameters is then

```
%% Pre-processing
% Three unit disks
O = [-5, 0, 5; -2, 0, 2];
a = [1, 1, 1];
%Set the parameters...
k = 1; %wavenumber
beta_inc = 0; %incident angle (plane wave case)
%Fourier series truncation parameter
M_modes = FourierTruncation(a, k, 'Min', 1);
```

For each integral equation, we now present the assembly process, the computation of the solution and finally the post-processing of the computed wave fields. The common pre-processing part is the one described above. All the functionalities presented here are also available in the file `BenchmarkDirichlet.m` which is located in the `Examples/Benchmark` folder.

4.1.2 The case of the EFIE

This integral formulation reads as

$$\left\{ \begin{array}{rcl} u & = & \mathscr{L}\rho, \\ L\rho & = & -u^{\mathrm{inc}}|_\Gamma, \end{array} \right.$$

where the first line is the integral equation representation of the exterior wavefield u and the second one is the surface integral equation to solve.

Dense storage

In μ-diff, the surface single-layer operator L and the incident plane wave field $u^{\mathrm{inc}}|_\Gamma$ are predefined quantities. If the full storage of the integral equation is used, then the direct solution of the resulting linear system can be obtained by the standard backslash Matlab operator \

```
%Right-hand side (plane wave)
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
%% Assembling
%Matrix of the system (the two following lines are the same)
L = SingleLayer(O, a, M_modes, k);
%% Solving (here, direct)
rho = L \ Uinc;
```

Sparse storage

For the sparse storage version, only the assembly process of the single-layer matrix and the system solution need to be modified as follows

```
%Matrix of the system (the two following lines are the same)
SpL = SpSingleLayer(O, a, M_modes, k);
%% Solving (here, direct)
rho = gmres(@(X)SpMatVec(X, M_modes, SpL), Uinc);
```

4.1.3 The case of the MFIE

The resolution of the scattering problem by the MFIE (1.16) leads to the integral equation representations

$$\left\{ \begin{array}{rcl} u & = & \mathscr{L}\rho, \\ \left(\dfrac{I}{2}+N\right)\rho & = & -\partial_{\mathbf{n}}u^{\mathrm{inc}}|_{\Gamma}. \end{array} \right.$$

Dense storage

The MFIE operator

$$\left(\frac{I}{2}+N\right)$$

can be computed thanks to the frontal function `IntegralOperator` with two arguments: the type of the operators (for the identity operator and the double-layer potential operator N, see Table 3.2) and their associated weights (0.5 and 1).

```
%Right hand side
DnUinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
%% Assembling
%Matrix of the system (the two following lines are the same)
A_MFIE = IntegralOperator(O, a, M_modes, k, [1, 4], [0.5, 1]);
%% Solving (here, direct)
rho = A_MFIE \ DnUinc;
```

The post-processing part is exactly the same as for the EFIE since the surface equation is based on the volume single-layer integral representation.

Sparse storage

The sparse storage version is almost the same as for the dense storage except for assembling the matrix and solving the linear system. Indeed, the matrices I and N cannot be computed by the same function since the sparse function representations `SpIntegralOperator` and `IntegralOperator` cannot be summed together. It is however possible to add the identity to a "sparse operator" thanks to `SpAddIdentity`

```
SpN = SpDnSingleLayer(O, a, M_modes, k);
%Add I/2 to N:
SpA_MFIE = SpAddIdentity(SpN, 0.5, M_modes)
rho = gmres(@(X)SpMatVec(X, M_modes, SpA_MFIE), DnUinc);
```

4.1.4 The case of the CFIE

Let us now consider the well-posed and well-conditioned CFIE (see also Eq. (1.19))

$$\begin{cases} u &= \mathscr{L}\rho, \\ \left[\alpha\eta L + (1-\alpha)\left(\frac{I}{2}+N\right)\right]\rho &= -\alpha\eta u^{\text{inc}}|_\Gamma - (1-\alpha)\partial_{\mathbf{n}}u^{\text{inc}}|_\Gamma. \end{cases}$$

Here, we fix the parameters to $\alpha = 0.5$ and $\eta = i/k$.

Dense storage

The operator

$$(1-\alpha)\left(\frac{I}{2}+N\right) + \alpha\eta L$$

is computed in μ-diff by using the `IntegralOperator` function, the post-processing remaining unchanged,

```
%CFIE
alpha = 0.5;
eta = i/k;
%Right-hand side
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
DnUinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
BCFIE = alpha*eta*Uinc + (1-alpha)*DnUinc;
%% Assembling
%Matrix of the system (the two following lines are the same)
ACFIE = IntegralOperator(O, a, M_modes, k, [2, 1, 4], [alpha*eta, ...
    0.5*(1-alpha), 1-alpha]);
%% Solving (here, direct)
rho = ACFIE \ BCFIE;
```

Sparse storage

The sparse storage version changes compared to the dense one: the operators $I/2 + N$ and L are computed separately and merged during the matrix-vector products. This is done in the `SpMatVec` function

```
SpL = SpSingleLayer(O, a, M_modes, k);
SpN = SpDnSingleLayer(O, a, M_modes, k);
SpA_MFIE = SpAddIdentity(SpN, 0.5, M_modes)
%% Solving and combining operators:
rho = gmres(@(X)SpMatVec(X,M_modes,{SpL, SpA_MFIE}, [alpha*eta, 1-alpha]), B_CFIE);
```

4.1.5 The case of the single-scattering preconditioned integral equation

We strongly recommend to use the single-scattering preconditioned version of the EFIE, which is rigorously the same as the MFIE and CFIE and, up to an invertible operator, to any other boundary integral equation (see Proposition 1.9). The EFIE version is available in μ-diff and is represented as

$$\begin{cases} u = \mathscr{L}\rho, \\ \widehat{L}^{-1}L\rho = -\widehat{L}^{-1}u^{\text{inc}}. \end{cases}$$

Dense storage

In μ-diff, the quantity $-\widehat{L}^{-1}u^{\text{inc}}|_\Gamma$ is provided by `PlaneWavePrecond` whereas $\widehat{L}^{-1}L$ is obtained with `PrecondDirichlet`. The syntax for the dense version is then the following

```
[...]
%Right-hand side
UincPrecond = PlaneWavePrecond(O, a, M_modes, k, beta_inc);
%Matrix of the system (the two following lines are the same)
APrecond = PrecondDirichlet(O, a, M_modes, k);
%Solving (here, directly)
rho = APrecond \ UincPrecond;
[...]
```

Sparse storage

The sparse storage is here almost the same as for the dense version thanks to `SpPrecondDirichlet`

```
SpPrecond = SpPrecondDirichlet(O, a, M_modes, k);
%% Solving and combining operators:
rho = gmres(@(X)SpMatVec(X,M_modes, SpPrecond), UincPrecond);
```

4.1.6 The case of the Brakhage-Werner integral equation

The Brakage-Werner integral equation for the Dirichlet problem (1.21) reads as

$$\begin{cases} u = (-\eta\mathscr{L} - \mathscr{M})\psi, \\ \left[-\eta L + \left(\dfrac{I}{2} - M\right)\right]\psi = -u^{\text{inc}}. \end{cases}$$

Dense storage

As in the previous chapter, we make here use of the common function `IntegralOperator`:

```
[...]
% eta parameter
eta_BW = i/k;
%Right-hand side
Uinc = PlaneWave(O, a, M_modes, k, beta_inc);
%Matrix of the system (the two following lines are the same)
A = IntegralOperator(O, a, M_modes, k, [1, 2, 3], [0.5, -eta_BW, -1]);
%Solving (here, directly)
rho = APrecond \ UincPrecond;
[...]
```

Sparse storage

For the sparse storage, the three operators I, L and N must be computed separately and merged during each matrix-vector products. Note that, in fact, the operators I and N can be merged together, as shown below using `SpAddIdentity`:

```
SpL = SpSingleLayer(O, a, M_modes, k); %L
SpM = -SpDoubleLayer(O, a, M_modes, k); %-M
SpIminusM = SpAddIdentity(SpM, 0.5, M_modes); %0.5I -M
%% Solving and combining operators:
psi_BW = gmres(@(X)SpMatVec(X,M_modes, {SpL, SpIminusM}, [-eta_BW, 1]), Uinc);
```

4.1.7 Post-processing

The EFIE, MFIE, the CFIE and their preconditioned version share the same integral representation of the scattered field u as a single-layer potential only: $u = \mathscr{L}\rho$. Their post-processing operations are the same. For the Brakhage-Werner integral equation, the post-processing is different since $u = (-\eta\mathscr{L} - \mathscr{M})\,\psi$.

Radar Cross Section (RCS) and far field

Once the surface wavefield has been computed, the far field can be calculated by the following μ-diff commands, here for a discretization of $[0, 2\pi[$ with a step of 1 degree:

```
%% Post-processing
%Scattering angles
theta_RCS = 0:360;
theta_RCS_rad = theta_RCS*2*pi/360;
%Farfield for the single-layer representation (<-> [1,0])
FSingleLayer = FarField(O, a, M_modes, k, theta_RCS_rad, rho, [1, 0]);
%Farfield for the BWIE (<-> [-eta_BW, -1])
FBWIE = FarField(O, a, M_modes, k, theta_RCS_rad, psi_BW, [-eta_BW, -1]);
```

The RCS can also be computed simply by:

```
RCSSingleLayer = FarFieldToRCS(FSingleLayer);
RCSBWIE = FarFieldToRCS(FBWIE);
```

or, if the far fields are not computed:

```
RCSSingleLayer = RCS(O, a, M_modes, k, theta_RCS_rad, rho, [1, 0]);
RCSBWIE = RCS(O, a, M_modes, k, theta_RCS_rad, psi_BW, [-eta_BW, -1]);
```

Near fields

The scattered field can also be computed on a grid. Even with a vectorization of the code, this functionality remains slow and cpu consuming. For the EFIE/MFIE and CFIE, only the single-layer potential must be computed whereas for the Brakhage-Werner integral equation, the double-layer must also be computed. In all the case, the computations are done in the exterior of the obstacles and `ExternalPotential` will do the job properly:

```
%% Building the grid
XXmin = -10; XXmax = 10;
YYmin = -10; YYmax = 10;
lc = 0.1;
XX = [XXmin:lc:XXmax];
YY = [YYmin:lc:YYmax];
```

```
[X,Y] = meshgrid(XX,YY);
%% Scattered field (single-layer potential)
U = ExternalPotential(X, Y, O, a, M_modes, k, rho, [1,0], 'OnBoundary', 1);
%% Scattered field (Brakhage-Werner integral equation)
UBWIE = ExternalPotential(X, Y, O, a, M_modes, k, psi_BW, [-eta_BW, -1], ...
    'OnBoundary', 1);
```

The total field $u_T = u + uinc$ can also be calculated, thanks to `IncidentWaveOnGrid` which computes the incident wave on a grid, even inside the obstacles (on these point, 0 value must be set, using `MaskMatrixObstacles`).

```
%% Incident wave
UincOnMesh = IncidentWaveOnGrid(X, Y, k, 'PlaneWave', beta_inc);
%Set 0 to points in obstacles
Matrix_Not_Obstacles = MaskMatrixObstacles(X, Y, O, a) == 0;
UincOnMesh = UincOnMesh.*Matrix_Not_Obstacles;
%% Total field
U_tot = U + UincOnMesh;
UBWIE_tot = UBWIE + UincOnMesh;
```

Note that, to display the near field, we suggest to use the following script, where the white artefacts are removed thanks to `set(gcf,'Renderer','Zbuffer');` and the circles are also displayed, using a high `zdata` (here, displaying a value called `U_tot`).

```
ind_fig = 1;
figure(ind_fig)
hold on;
surf(X,Y, abs(U_tot));
shading interp;
view(2); colorbar;
PlotCircles(O, a, ind_fig, 'Color', 'k', 'LineWidth', 2, 'zdata', ...
    max(max(abs(U_tot))));
set(gcf,'Renderer','Zbuffer');
hold off
```

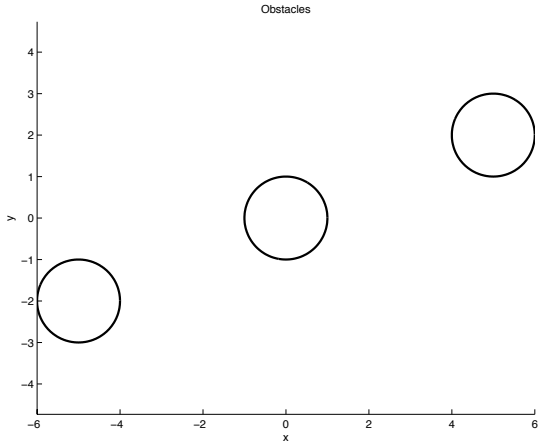
4.1.8 Results

The figure 4.1 presents the results obtained with the considered configuration, that is 3 unit disks placed on $(-5,-2)$, $(0,0)$ and $(5,2)$ with $k = 1$ and an incident plane wave of direction 0. On the figure is shown the obstacles, the history of convergence of the GMRES for the 5 integral equations for their dense and sparse storage, the radar cross section and also the near-fields.

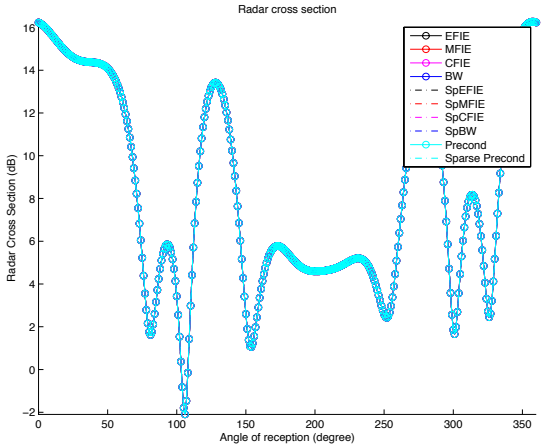
4.1.9 Point source wave

Our toolbox μ-diff also provides a right hand side derived from a wave emitted by a point source. To solve that kind of problem, the right hand side is built thanks to `PointSource` and `DnPointSource` instead of respectively `PlaneWave` and `DnPlaneWave` (there is currently no equivalent of `PlaneWavePrecond`). For example, for the EFIE and a point source centered on $(-10,0)$, the right hand side must be computed as:

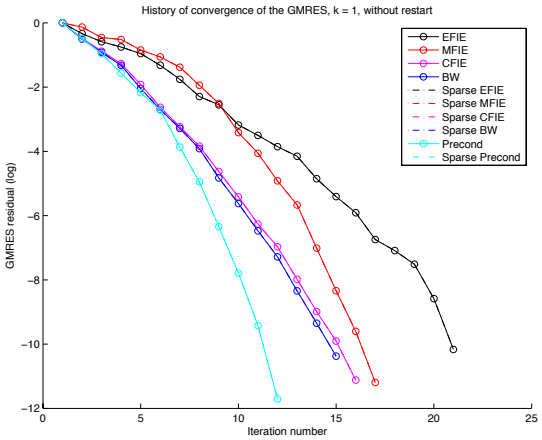
```
XS = [-10; 0]; %location of the source (point source case)
%Right-hand side
```



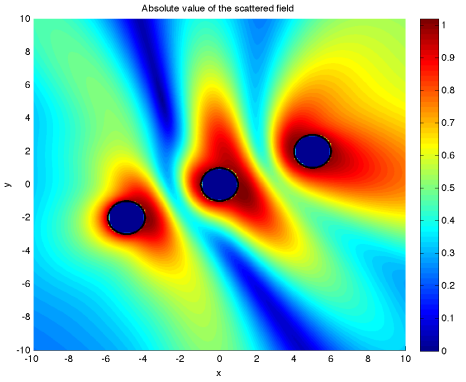
(a) Obstacles



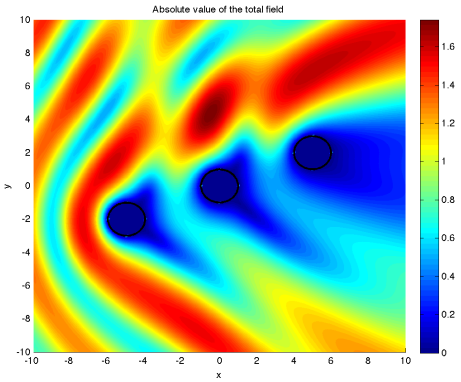
(b) Radar cross section



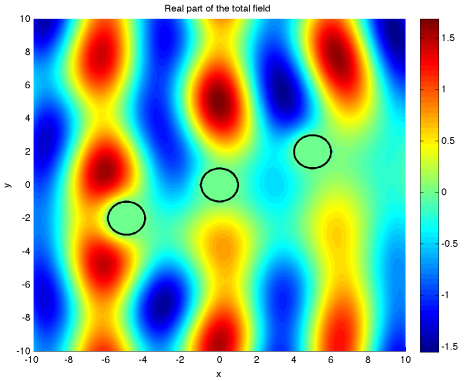
(c) History of convergence of the GMRES



(d) Absolute value of the scattered field



(e) Absolute value of the total field



(f) Real part of the total field

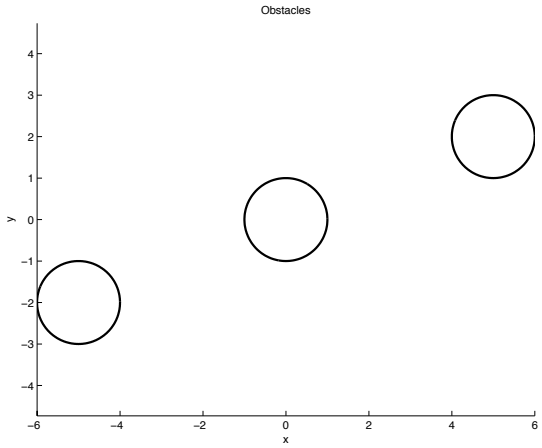
Figure 4.1: Different results for a Dirichlet problem and an incident plane wave solved using μ-diff. The first figure shows the three obstacles and the second one the radar cross section obtained for the different integral equation (all superimposed). The next figure (c) represents the GMRES history of convergence of the different integral equations. The figure (d) shows the absolute value of the scattered field, and the last two, (e) and (f), represent respectively the absolute value and the real part of the total field.

```
Uinc = PointSource(O, a, M_modes, k, XS);
```

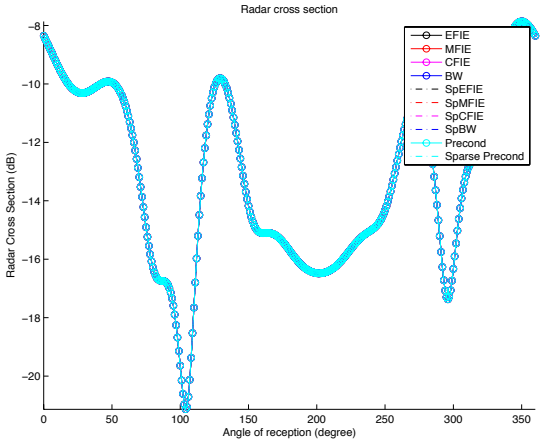
The other change appears in the post-processing when computing the total field on a grid. Indeed, the incident wave is different and `IncidentWaveOnGrid` must get the right argument:

```
UincOnMesh = IncidentWaveOnGrid(X, Y, k, 'PointSource', XS);
```

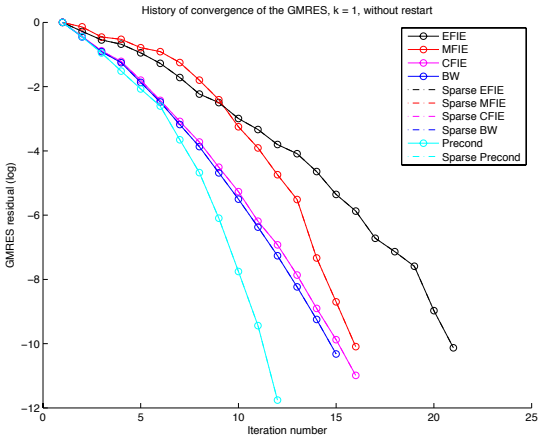
The rest of the code remains unchanged and the results obtained with a point source are shown on figure 4.2.



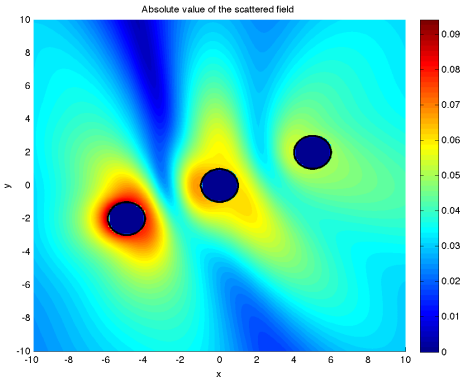
(a) Obstacles



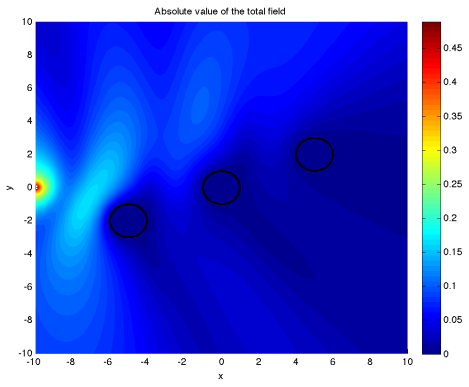
(b) Radar cross section



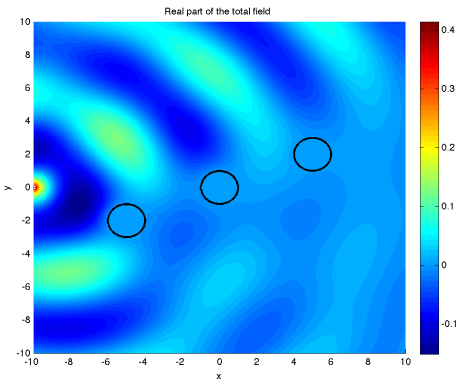
(c) History of convergence of the GMRES



(d) Absolute value of the scattered field



(e) Absolute value of the total field



(f) Real part of the total field

Figure 4.2: Different results for a Dirichlet problem for a wave emitted by a point source solved using μ-diff. The first two pictures show respectively the obstacles and the radar cross section, while the third one shows the GMRES history of convergence. The absolute value of the scattered field is presented on subfigure (d), and the total field on figures (e) (absolute value) and (f) (real part).

4.2 The Neumann boundary-value problem

Let us now consider the sound-hard scattering problem

$$\begin{cases} (\Delta + k^2)u &= 0, & \text{in } \Omega^+, \\ \partial_{\mathbf{n}} u &= -\partial_{\mathbf{n}} u^{\text{inc}}, & \text{on } \Gamma, \\ & u \text{ outgoing.} \end{cases}$$

An efficient solution to this problem is given for example by a preconditioned integral equation for sound-hard obstacles. Here, we only present this solution but the extension to other kinds of integral equations is direct. The μ-diff script is close to the one developed for the Dirichlet problem, only the two following functions must be modified: `PrecondDirichlet` is replaced by `PrecondNeumann` and the right-hand side `PlaneWavePrecond` is now given by `DnPlaneWavePrecond`. For the Neumann problem, the preconditioned boundary integral equation is based on the double-layer representation

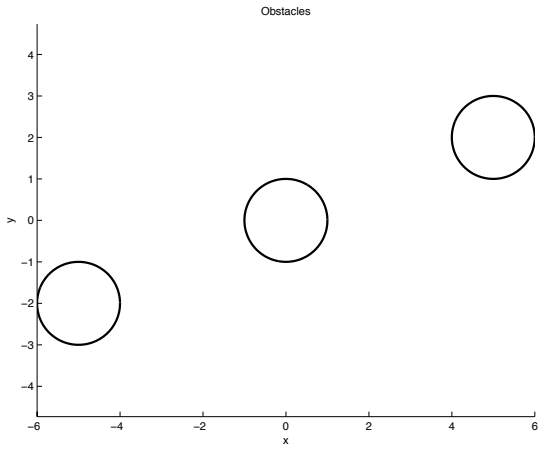
$$\begin{cases} u = \mathscr{M}\lambda, \\ \widehat{D}^{-1} D\lambda = -\widehat{D}^{-1}\partial_{\mathbf{n}} u^{\text{inc}}. \end{cases}$$

```
% Three unit disks
O = [-5, 0, 5; -2, 0, 2];
a = [1, 1, 1];
%Set the parameters...
k = 1; %wavenumber
beta_inc = 0; %incident angle
%Fourier series truncation parameter
M_modes = FourierTruncation(a, k, 'Min', 1);
%Right-hand side
DnUincPrecond = DnPlaneWavePrecond(O, a, M_modes, k, beta_inc);
%Matrix of the system (the two following lines are the same)
APrecond = PrecondNeumann(O, a, M_modes, k);
%Solving (here, direct)
lambda = APrecond \ DnUincPrecond;
```

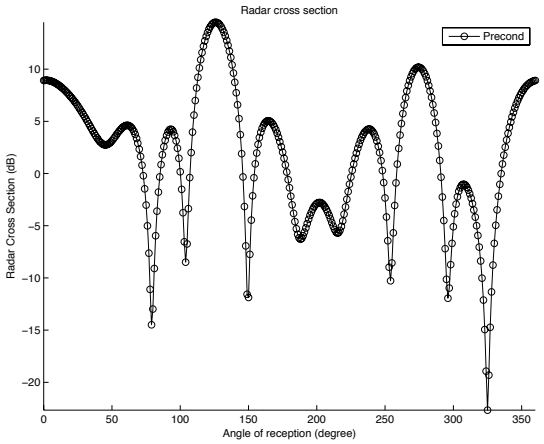
The post-processing is based on the double-layer potential (compared to Dirichlet, the modification is realized in the `RCS` function, the last argument `[1,0]` is then replaced by `[0,1]`)

```
%Scattering angles
theta_RCS = 0:360;
theta_RCS_rad = theta_RCS*2*pi/360;
%Radar Cross Section associated with the double-layer potential
myRCS = RCS(O, a, M_modes, k, theta_RCS_rad, lambda, [0,1]);
plot(theta_RCS, myRCS, 'k');
```

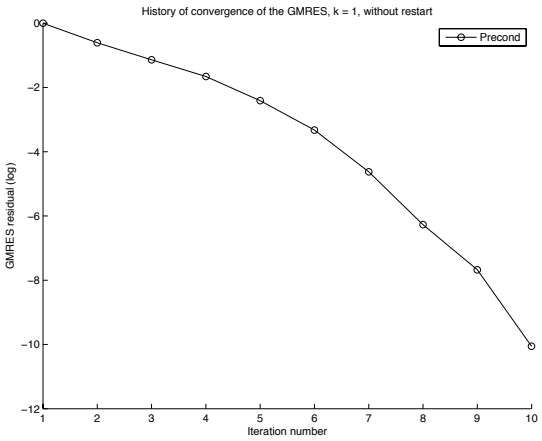
With the previous parameter, we obtain the result shown on figure 4.3 for an incident plane wave of direction 0 and on figure 4.4 for a point source centered on $[-10, 0]$. Only the case of the preconditioned integral equation has been shown.



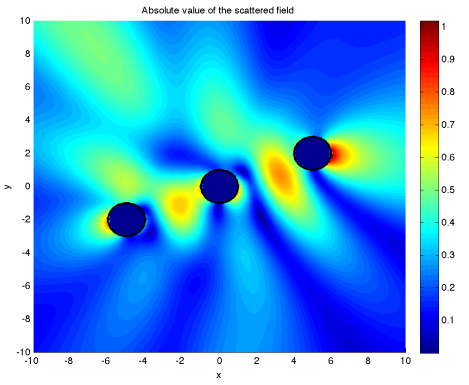
(a) Obstacles



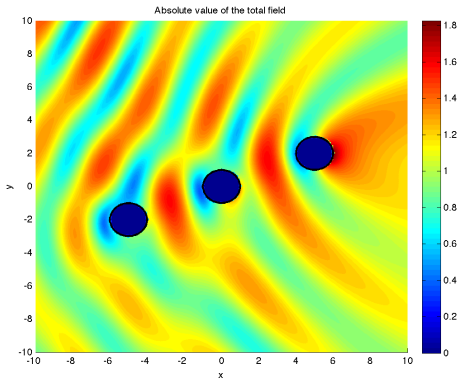
(b) Radar cross section



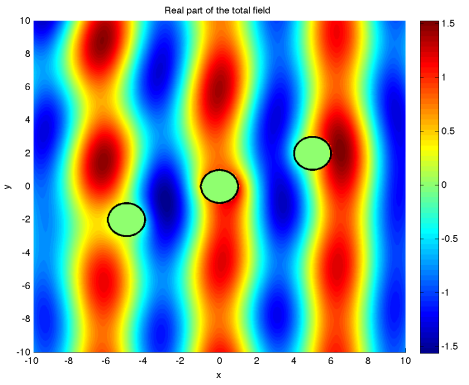
(c) History of convergence of the GMRES



(d) Absolute value of the scattered field

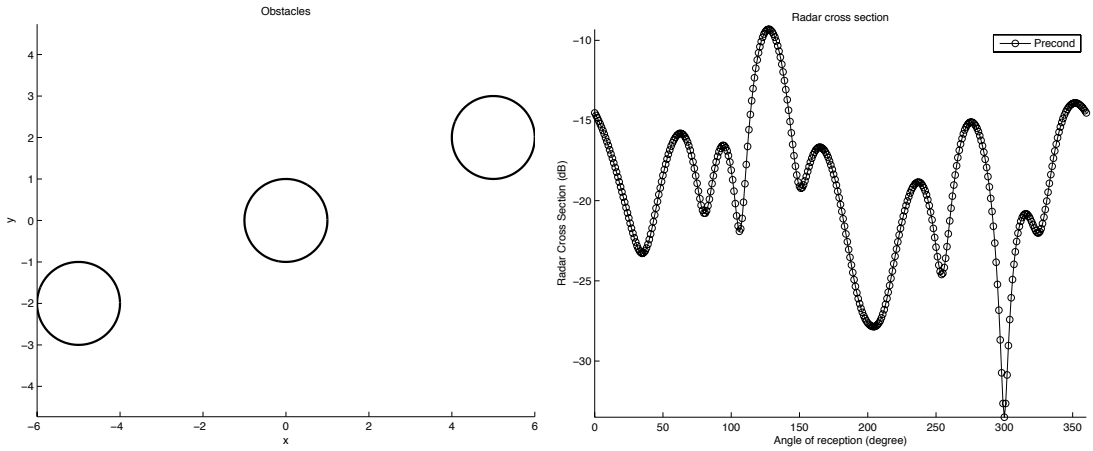


(e) Absolute value of the total field



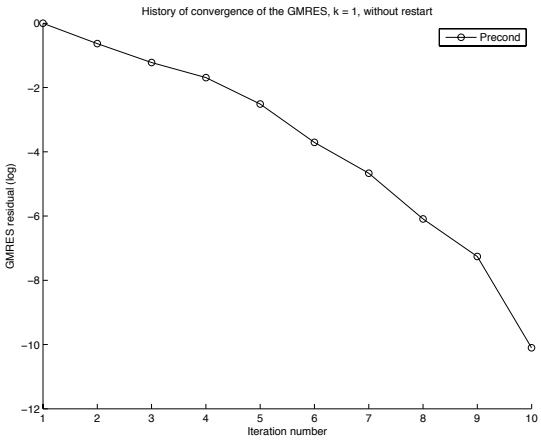
(f) Real part of the total field

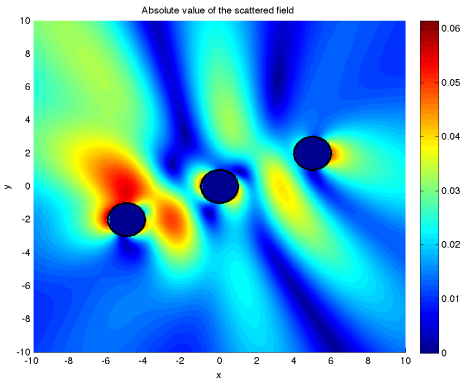
Figure 4.3: Neumann boundary value problem with an incident plane wave of direction 0, solved using (only) the single scattering preconditioned integral equation. The three first figures show respectively: the obstacles, the radar cross section, the history of convergence of the GMRES. The three last, (d) to (f), present the absolute value of the scattered field, the absolute and real part of the total field.



(a) Obstacles

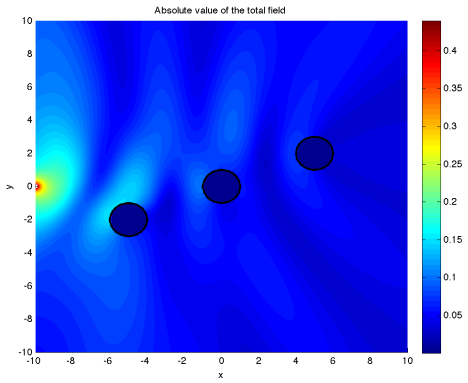
(b) Radar cross section

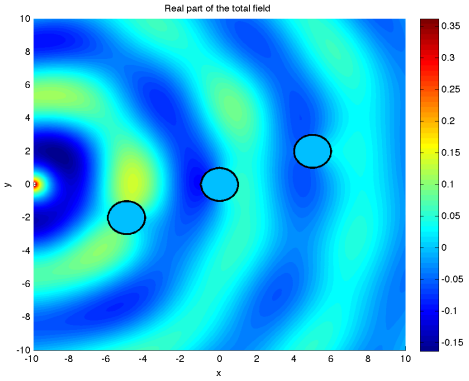




(c) History of convergence of the GMRES

(d) Absolute value of the scattered field





(e) Absolute value of the total field

(f) Real part of the total field

Figure 4.4: Neumann boundary value problem with a point source emitter solved using the single scattering preconditioned integral equation. In the respective order is shown: the obstacles, the radar cross section, the history of convergence of the GMRES and then, the absolute value of the scattered field, the absolute and real part of the total field.

4.3 Mixing Dirichlet and Neumann boundary conditions

Let us consider the following situation where we mix Dirichlet and Neumann boundary conditions. The scatterer is composed of M_D sound-soft and $M_N = M - M_D$ sound-hard obstacles, leading to the scattering problem

$$\begin{cases} (\Delta + k^2)u &= 0, & \text{in } \Omega^+, \\ u &= -u^{\text{inc}}, & \text{on } \Gamma_p, \quad p = 1, \dots, M_D, \\ \partial_{\mathbf{n}} u &= -\partial_{\mathbf{n}} u^{\text{inc}}, & \text{on } \Gamma_p, \quad p = M_D + 1, \dots, M, \\ & u \text{ outgoing.} \end{cases}$$

For this problem, the preconditioned integral is not directly available. We can apply a Combined Field Integral Equation for the mixed problem (see equation (1.34)) and written as

$$(\frac{I}{2} + A)\varphi = b,$$

where the matrix A is given by (1.35). We have

$$A(p,q) = \begin{cases} (1-\alpha)N^{p,q} + \alpha\eta L^{p,q}, & \text{if } q \leq M_D, \\ (1-\alpha)D^{p,q} + \alpha\eta M^{p,q}, & \text{if } q > M_D. \end{cases}$$

The matrix is particularly easy to build with μ-diff thanks to the frontal function `IntegralOperator`. To this end, two three-dimensional arrays, `Assembling` and `Weight`, are built such that

$$\texttt{Assembling}(:,p,q) = \begin{cases} [4,2], & \text{if } q \leq M_D, \\ [5,3], & \text{if } q > M_D, \end{cases} \quad \text{and} \quad \texttt{Weight}(:,p,q) = [(1-\alpha), \alpha\eta].$$

The indices in `Assembling` corresponds to the indices of the boundary integral operators ($2 = L^{p,q}$, $3 = M^{p,q}$, $4 = N^{p,q}$, $5 = D^{p,q}$). The assembling process is realized by `IntegralOperator`.

```
% Two Dirichlet obstacles (unit diks)
OD = [-5, -5; -5, 5];
aD = [1, 1];
N_scatD = length(aD);
% Two Neumann obstacles (unit diks)
ON = [5, 5; -5, 5];
aN = [1, 1];
N_scatN = length(aN);
%All obstacles
O = [OD, ON];
a = [aD, aN];
N_scat = N_scatD + N_scatN;
%Set the parameters...
k = 1; %wavenumber
beta_inc = 0; %incident angle
%Fourier series truncation parameter (Dirichlet, Neumann, All)
M_modesD = FourierTruncation(aD, k, 'Min', 1);
M_modesN = FourierTruncation(aN, k, 'Min', 1);
M_modes = [M_modesD, M_modesN];
%Right-hand side
Uinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
DnUinc = DnPlaneWave(O, a, M_modes, k, beta_inc);
B = alpha*eta*Uinc + (1-alpha)*DnUinc;
%% Assembling
Assembling = zeros(2, N_scat, N_scat);
```

```
Weight = zeros(2, N_scat, N_scat);
for p=1:N_scatD
    for q=1:N_scat
        Assembling(:,p,q) = [4;2];
        Weight(:,p,q) = [1-alpha; alpha*eta];
    end
end
for p=N_scatD+1:N_scat
    for q=1:N_scat
        Assembling(:,p,q) = [5;3];
        Weight(:,p,q) = [1-alpha; alpha*eta];
    end
end
%Common function
A = IntegralOperator(O, a, M_modes, k, Assembling, Weight);
%Solving (here, direct)
density = A \ B;
```

The post-processing is then done by specifying to μ-diff how the density must be used: for the first M_D obstacles, a single-layer potential is used, while for the others, a double-layer potential is required. The `RCS` function can simply do that. It just needs an array `TypeOfOp` of size `N_scat` $\times$ 2 such that

$$\texttt{TypeOfOp}(p,:) = \begin{cases} [1,0], & \text{if } p \leq M_D, \\ [0,1], & \text{if } p > M_D. \end{cases}$$

In a μ-diff script, this means "Apply the single-layer potential (multiplied by 1) for the first M_D part of the density and a double-layer potential (multiplied by 1) for the others".

```
%Preparing TypeOfOp
TypeOfOp = zeros(N_scat, 2);
for p =1:N_scat
    if(p ≤ N_scatD)
        TypeOfOp(p,1) = 1;
    else
        TypeOfOp(p,2) = 1;
    end
end
%Scattering angles
theta_RCS = 0:360;
theta_RCS_rad = theta_RCS*2*pi/360;
%Radar Cross Section computation
myRCS = RCS(O, a, M_modes, k, theta_RCS_rad, density, TypeOfOp);
plot(theta_RCS, myRCS, 'k');
```

4.4 Penetrable case

4.4.1 Integral equation

Let us now consider the case of penetrable obstacles, with a frequency k_p^- (possibly different) in each scatterer Ω_p^-, for $p = 1, \ldots, M$. The problem then reads as:

$$\left\{\begin{array}{rcll} (\Delta + (k^+)^2)u^+ & = & 0, & \text{in } \Omega^+, \\ (\Delta + (k^-)^2)u^- & = & 0, & \text{in } \Omega^-, \\ u^+ - u^- & = & -u^{\text{inc}}, & \text{on } \Gamma, \\ \partial_{\mathbf{n}} u^+ - \partial_{\mathbf{n}} u^- & = & -\partial_{\mathbf{n}} u^{\text{inc}}, & \text{on } \Gamma, \\ u^+ \text{ outgoing}, & & & \end{array}\right.$$

where $k^-|_{\Omega_p^-} = k_p^-$. Solving this problem can be done thanks to the integral equation presented in section 1.36 where the interior total field u^- (equal also to the total field) and the exterior scattered field u^+ are represented as a single-layer potential:

$$\left\{\begin{array}{ll} u^+(\mathbf{x}) = \mathscr{L}^+\rho^+(\mathbf{x}) = \displaystyle\int_\Gamma G^+(\mathbf{x},\mathbf{y})\rho^+(\mathbf{y})\,\mathrm{d}\mathbf{y} & \text{in } \Omega^+, \\ u^-(\mathbf{x}) = \mathscr{L}^-\rho^-(\mathbf{x}) = \displaystyle\int_\Gamma G^-(\mathbf{x},\mathbf{y})\rho^-(\mathbf{y})\,\mathrm{d}\mathbf{y} & \text{in } \Omega^-, \end{array}\right.$$

The associated integral equation (1.36) is recalled to be

$$\begin{pmatrix} L^+ & -L^- \\ -\dfrac{I}{2} + N^+ & -\dfrac{I}{2} - N^- \end{pmatrix} \begin{pmatrix} \rho^+ \\ \rho^- \end{pmatrix} = \begin{pmatrix} -u^{\text{inc}}|_\Gamma \\ -\partial_{\mathbf{n}} u^{\text{inc}}|_\Gamma \end{pmatrix}.$$

4.4.2 A more complex geometry

As already explained in §3.3.1, μ-diff also proposes some tools to place the obstacles and in a particular order, leading to original geometry, as the one shown previously on figure 3.3 and treated here (see 4.5(a)). The geometry is composed by a periodic placement of 11×11 unit disks, separated by a distance equal to 1 with an empty middle line and middle column, leading to, in fact, $10 \times 10 = 100$ obstacles. Below is recalled the code to obtain this geometry:

```
bx = 3; by = 3;
Nx = 11; Ny = 11;
O = RectangularLattice(bx, by, Nx, Ny, 'Centered', [0, 0]);
a = ones(1, size(O, 2));
[O, a] = RemoveDisk(O, a, 'X', 0, 'Y', 0);
N_scat = size(O,2);
```

4.4.3 Writing and solving the BIE using μ-diff

Let us solve the problem using a dense storage and first write the matrix of the system, which consists in assembling the different operator into a larger matrix:

```
Splus = SingleLayer(O, a, M_modes, k);
Sminus = IntegralOperator(O, a, M_modes, k_int, 2*eye(N_scat,N_scat));
Nplus = DnSingleLayer(O, a, M_modes, k);
Nminus = IntegralOperator(O, a, M_modes, k_int, 4*eye(N_scat,N_scat));
Identity = eye(size(Nplus));
```

```
% Boundary integral operator
A = [Splus, -Sminus; -0.5*Identity + Nplus, -(0.5*Identity + Nminus)];
%cleaning memory
clear Splus Sminus Nplus Nminus Identity mu_matrix;
```

Second, we need to introduce the right hand side (here a point source):

```
Uinc = PointSource(O, a, M_modes, k, XS);
DnUinc = DnPointSource(O, a, M_modes, k, XS);
[B] = [Uinc;DnUinc];
clear Uinc DnUinc;
```

Finally, we solve the system, here using a direct solver. The two densities ρ^+ and ρ^- are here also extracted to simplify:

```
rho = A\B;
%% Extracting the "exterior" and "interior" densities
rho_plus = rho(1:sum_modes);
rho_minus = rho(sum_modes+1:end);
clear rho;
```

4.4.4 Post-processing

Radar cross section

We compute now the radar cross section, which is based on the (exterior) single-layer potential:

```
theta_RCS = 0:360;
theta_RCS_rad = theta_RCS*2*pi/360;
R = RCS(O, a, M_modes, k, theta_RCS_rad, rho_plus, [1,0]);
```

The radar cross section is shown on figure 4.5(b).

Near field

Let us now compute the near-fields by first building the grid:

```
XX = [-20:0.1:20];
YY = [-20:0.1:20];
[X,Y] = meshgrid(XX,YY);
```

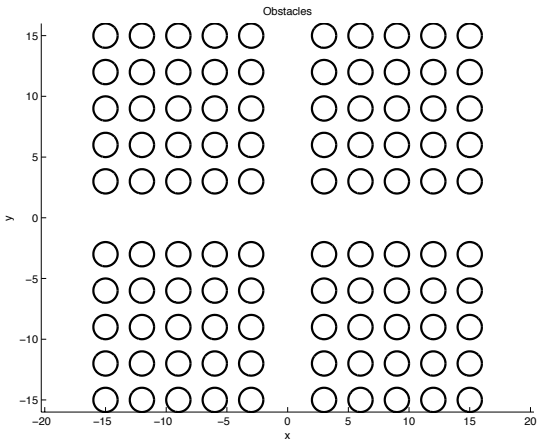
And second, we compute the single-layer potentials, external and internal:

```
%%Potentials
Ue = ExternalPotential(X, Y, O, a, M_modes, k, rho_plus, [1,0]);
Ui = InternalPotential(X, Y, O, a, M_modes, k_int, rho_minus, [1,0], ...
    'OnBoundary', 1);
U = Ue + Ui;
```

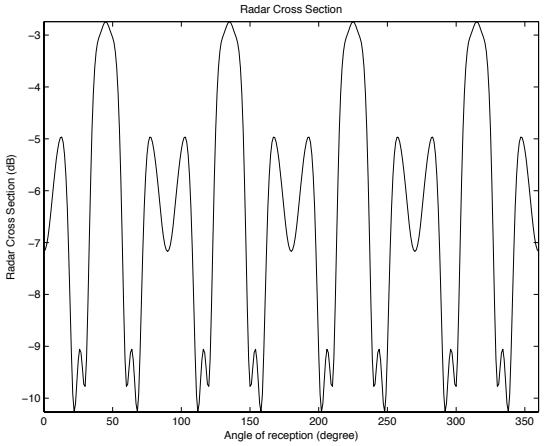
The quantity `U` is the scattered field outside the obstacles and the total field inside in the obstacles. The option `OnBoundary` is set to compute the potential also on the boundary, in case there is some points of the grid on it. To get the total field, we need to add the incident wave to `U`, outside the obstacles:

```
UincOnMesh = IncidentWaveOnGrid(X, Y, k, 'PointSource', XS);
Matrix_Not_Obstacles = (MaskMatrixObstacles(X, Y, O, a) == 0);
UincOnMesh = UincOnMesh.*Matrix_Not_Obstacles;
U_tot = U + UincOnMesh;
```

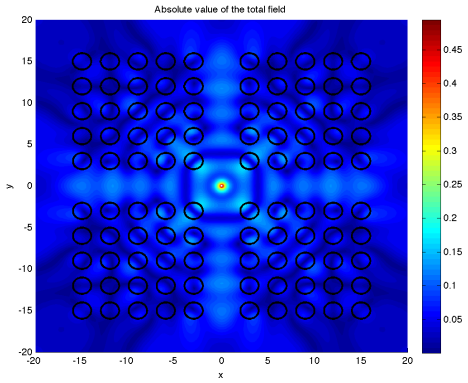
The near-fields are now fully computed and can be displayed. Figure 4.5(c) represents the absolute value of the total field, and 4.5(d) and 4.5(e) its real and imaginary part.



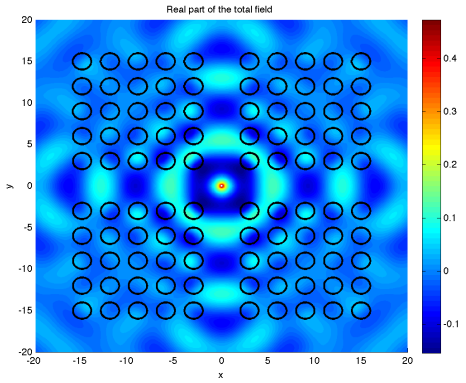
(a) Obstacles



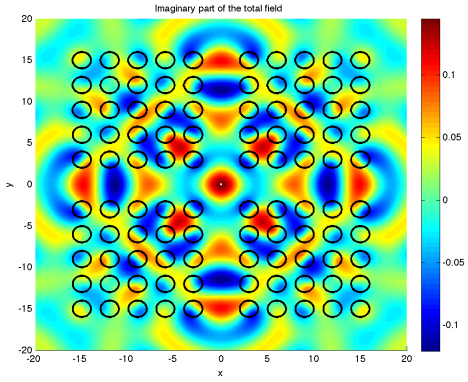
(b) Radar cross section



(c) Absolute value of the total field



(d) Real part of the total field



(e) Imaginary part of the total field

Figure 4.5: Results obtained for a complex geometry of penetrable obstacles with $k = 1$ and $k^- = 2k$, solved using a single-layer potential representation of the field. The point source is located on $(0, 0)$, in the center of the geometry.

Appendix A

List of the μ-diff functions (alphabetical order)

- `BenchmarkCalderon` (*Examples/Benchmark*)
Example of solution of penetrable scattering using Calderon projectors
- `BenchmarkDirichlet` (*Examples/Benchmark*)
Example of solution of sound soft scattering using different integral equations
- `BenchmarkNeumann` (*Examples/Benchmark*)
Example of solution of sound hard scattering using different integral equations
- `BenchmarkPenetrable` (*Examples/Benchmark*)
Example of solution of penetrable scattering using single layer potential
- `BlockCalderonProjector` (*IntOperators/Dense/Interface/Block*)
Calderon projector block
- `BlockDnDoubleLayer` (*IntOperators/Dense/Interface/Block*)
Block of the normal derivative of the double layer integral operator
- `BlockDnPlaneWave` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the normal derivative of a plane wave
- `BlockDnPlaneWavePrecond` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the normal derivative of a plane wave for the preconditioned problem of sound-hard scattering
- `BlockDnPointSource` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the normal derivative of a point source wave
- `BlockDnSingleLayer` (*IntOperators/Dense/Interface/Block*)
Block of the normal derivative of the single layer integral operator
- `BlockDoubleLayer` (*IntOperators/Dense/Interface/Block*)
Block of the double layer integral operator
- `BlockIdentity` (*IntOperators/Dense/Interface/Block*)
Block of the identity integral operator
- `BlockIncidentWave` (*PreProcessing/IncidentWave*)
Block vector of a generic incident wave (right hand side, block=1 obstacle)

- `BlockIntegralOperator` (*IntOperators/Dense*)
Generic dense block of an integral operator
- `BlockPlaneWave` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the trace of a plane wave
- `BlockPlaneWavePrecond` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the trace of a plane wave for the preconditioned problem of sound-soft scattering
- `BlockPointSource` (*PreProcessing/IncidentWave/Block*)
Block vector (=1 obstacle) of a the right hand side of the trace of a point source wave
- `BlockPotential` (*PostProcessing/NearField/Functions*)
Generic block potential matrix used to compute external or internal potentials
- `BlockPrecondDirichlet` (*IntOperators/Dense/Interface/Block*)
Block of the preconditioned integral operator (sound soft case)
- `BlockPrecondNeumann` (*IntOperators/Dense/Interface/Block*)
Block of the preconditioned integral operator (sound hard case)
- `BlockSingleLayer` (*IntOperators/Dense/Interface/Block*)
Block of the single layer integral operator
- `BoundaryOfObstacles` (*PostProcessing/Geometry*)
Extract the boundary of the obstacle
- `CalderonProjector` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the Calderon projector
- `CheckPlacement` (*PreProcessing/Geometry*)
Verify if the obstacles are satisfying the condition (overlapping, ...)
- `CreateRandomDisks` (*PreProcessing/Geometry*)
Place randomly random disks in a box
- `dbesselh` (*Common*)
First derivative of first kind Hankel function
- `dbesselj` (*Common*)
First derivative of Bessel function
- `dbessely` (*Common*)
First derivative of Newton function
- `DnDoubleLayer` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the normal derivative of the double layer integral operator
- `DnPlaneWave` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the normal derivative of a plane wave
- `DnPlaneWavePrecond` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the normal derivative of a plane wave for the preconditioned problem of sound-hard scattering
- `DnPointSource` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the normal derivative of a point source wave
- `DnSingleLayer` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the normal derivative of the single layer integral operator

- `DORT_dielectric` (*Examples/TimeReversal/FarField/NonPenetrable*)
DORT for penetrable obstacles (far field)
- `DORT_NonPenetrable` (*Examples/TimeReversal/FarField/NonPenetrable*)
DORT for acoustic sound soft obstacles (far field)
- `DoubleLayer` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the double layer integral operator
- `ExternalDoubleLayerPotential` (*PostProcessing/NearField/Interface*)
External potential of double layer potential only
- `ExternalPotential` (*PostProcessing/NearField*)
Compute potentials (single, double or linear combination) on a (Matlab) meshgrid and outside the obstacles
- `ExternalSingleLayerPotential` (*PostProcessing/NearField/Interface*)
External potential of single layer potential only
- `fangle` (*Common*)
Angle with horizontal axis
- `FarField` (*PostProcessing/FarField*)
Generic far field computation from densities
- `FarField_to_RCS` (*PostProcessing/FarField*)
Radar Cross Section (RCS) from far field
- `FarFieldDoubleLayer` (*PostProcessing/FarField/Interface*)
Far field of the double layer potential only
- `FarFieldSingleLayer` (*PostProcessing/FarField/Interface*)
Far field of the single layer potential only
- `FourierTruncation` (*PreProcessing/Fourier*)
Provide the number of mode to kept in the Fourier series
- `GetPotentialOptions` (*PostProcessing/NearField/Functions*)
Options for potential computations are condensed here
- `HerglotzWave` (*Examples/TimeReversal/FarField/Common*)
Compute an Herglotz wave (linear combination of plane waves)
- `Identity` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the identity operator
- `IncidentWave` (*PreProcessing/IncidentWave*)
Full vector of a generic incident wave (right hand side)
- `IncidentWaveOnGrid` (*PostProcessing/IncidentWave*)
Compute incident wave on a (Matlab) meshgrid
- `IntegralOperator` (*IntOperators/Dense*)
Generic integral operator dense matrix (full)
- `InternalDoubleLayerPotential` (*PostProcessing/NearField/Interface*)
Internal potential of double layer potential only
- `InternalPotential` (*PostProcessing/NearField*)
Compute potentials (single, double or linear combination) on a (Matlab) meshgrid and inside the obstacles

- `InternalSingleLayerPotential` (*PostProcessing/NearField/Interface*)
Internal potential of single layer potential only
- `MaskMatrixObstacles` (*PostProcessing/Geometry*)
Matrix with boolean values inside or outside obstacles
- `PlaneWave` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the trace of a plane wave
- `PlaneWavePrecond` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the trace of a plane wave for the preconditioned problem of sound-soft scattering
- `PlotCircles` (*PostProcessing/Geometry*)
Display obstacles on figure
- `PointSource` (*PreProcessing/IncidentWave/Full*)
Full vector (=all obstacle) of a the right hand side of the trace of a point source wave
- `PrecondDirichlet` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the preconditioned integral operator (sound soft case)
- `PrecondNeumann` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the preconditioned integral operator (sound hard case)
- `RCS` (*PostProcessing/FarField*)
Generic Radar Cross Section (RCS) computation from densities
- `RCSDoubleLayer` (*PostProcessing/FarField/Interface*)
Radar Cross Section (RCS) of the double layer potential only
- `RCSSingleLayer` (*PostProcessing/FarField/Interface*)
Radar Cross Section (RCS) of the single layer potential only
- `RectangularLattice` (*PreProcessing/Geometry*)
Build a rectangular lattice of disks
- `RemoveDisk` (*PreProcessing/Geometry*)
Remove some disks
- `repeat_horiz` (*Common*)
Copy/paste a row vector to build a matrix
- `repeat_vert` (*Common*)
Copy/paste a column vector to build a matrix
- `SingleLayer` (*IntOperators/Dense/Interface/Full*)
Full dense matrix of the single layer integral operator
- `SpAddIdentity` (*IntOperators/Sparse/Functions*)
Sparse function: add identity to a sparse operator
- `SpBlockDnDoubleLayer` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the normal derivative of the double layer integral operator
- `SpBlockDnSingleLayer` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the normal derivative of the single layer integral operator
- `SpBlockDoubleLayer` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the double layer integral operator
- `SpBlockIdentity` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the identity operator

- `SpBlockIntegralOperator` (*IntOperators/Sparse*)
Generic sparse block of an integral operator
- `SpBlockPrecondDirichlet` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the preconditioned integral operator (sound soft case)
- `SpBlockPrecondNeumann` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the preconditioned integral operator (sound hard case)
- `SpBlockSingleLayer` (*IntOperators/Sparse/Interface/Block*)
Sparse block of the single layer integral operator
- `SpDnDoubleLayer` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the normal derivative of the double layer integral operator
- `SpDnSingleLayer` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the normal derivative of the single layer integral operator
- `SpDoubleLayer` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the double layer integral operator
- `SpIdentity` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the identity operator
- `SpIntegralOperator` (*IntOperators/Sparse*)
Generic integral operator sparse matrix (full)
- `SpMatVec` (*IntOperators/Sparse/Functions*)
Sparse function: sparse matrix - vector product (possibly multiples)
- `SpPrecondDirichlet` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the preconditioned integral operator (sound soft case)
- `SpPrecondNeumann` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the preconditioned integral operator (sound hard case)
- `SpSingleLayer` (*IntOperators/Sparse/Interface/Full*)
Sparse matrix of the single layer integral operator
- `SpSingleMatVec` (*IntOperators/Sparse/Functions*)
Sparse function: sparse matrix - (only one) vector product
- `TimeReversalOperator` (*Examples/TimeReversal/FarField/Common*)
Time reversal matrix in acoustic and far field context
- `TriangularLattice` (*PreProcessing/Geometry*)
Build a triangular lattice of disks

Appendix B

List of the μ-diff functions (ordering by folder name)

Common

- `repeat_horiz`: Copy/paste a row vector to build a matrix
- `fangle`: Angle with horizontal axis
- `dbesselj`: First derivative of Bessel function
- `dbessely`: First derivative of Newton function
- `dbesselh`: First derivative of first kind Hankel function
- `repeat_vert`: Copy/paste a column vector to build a matrix

Examples/Benchmark

- `BenchmarkCalderon`: Example of solution of penetrable scattering using Calderon projectors
- `BenchmarkDirichlet`: Example of solution of sound soft scattering using different integral equations
- `BenchmarkNeumann`: Example of solution of sound hard scattering using different integral equations
- `BenchmarkPenetrable`: Example of solution of penetrable scattering using single layer potential

Examples/TimeReversal/FarField/Common

- `HerglotzWave`: Compute an Herglotz wave (linear combination of plane waves)
- `TimeReversalOperator`: Time reversal matrix in acoustic and far field context

Examples/TimeReversal/FarField/NonPenetrable

- `DORT_dielectric`: DORT for penetrable obstacles (far field)
- `DORT_NonPenetrable`: DORT for acoustic sound soft obstacles (far field)

IntOperators/Dense

- `BlockIntegralOperator`: Generic dense block of an integral operator
- `IntegralOperator`: Generic integral operator dense matrix (full)

IntOperators/Dense/Interface/Block

- `BlockDnSingleLayer`: Block of the normal derivative of the single layer integral operator
- `BlockDoubleLayer`: Block of the double layer integral operator
- `BlockIdentity`: Block of the identity integral operator
- `BlockPrecondDirichlet`: Block of the preconditioned integral operator (sound soft case)
- `BlockPrecondNeumann`: Block of the preconditioned integral operator (sound hard case)
- `BlockSingleLayer`: Block of the single layer integral operator
- `BlockDnDoubleLayer`: Block of the normal derivative of the double layer integral operator
- `BlockCalderonProjector`: Calderon projector block

IntOperators/Dense/Interface/Full

- `DnDoubleLayer`: Full dense matrix of the normal derivative of the double layer integral operator
- `DnSingleLayer`: Full dense matrix of the normal derivative of the single layer integral operator
- `DoubleLayer`: Full dense matrix of the double layer integral operator
- `SingleLayer`: Full dense matrix of the single layer integral operator
- `CalderonProjector`: Full dense matrix of the Calderon projector
- `PrecondNeumann`: Full dense matrix of the preconditioned integral operator (sound hard case)
- `PrecondDirichlet`: Full dense matrix of the preconditioned integral operator (sound soft case)
- `Identity`: Full dense matrix of the identity operator

IntOperators/Sparse

- `SpBlockIntegralOperator`: Generic sparse block of an integral operator
- `SpIntegralOperator`: Generic integral operator sparse matrix (full)

IntOperators/Sparse/Functions

- `SpAddIdentity`: Sparse function: add identity to a sparse operator
- `SpMatVec`: Sparse function: sparse matrix - vector product (possibly multiples)
- `SpSingleMatVec`: Sparse function: sparse matrix - (only one) vector product

IntOperators/Sparse/Interface/Block

- `SpBlockDoubleLayer`: Sparse block of the double layer integral operator
- `SpBlockPrecondNeumann`: Sparse block of the preconditioned integral operator (sound hard case)
- `SpBlockPrecondDirichlet`: Sparse block of the preconditioned integral operator (sound soft case)
- `SpBlockDnSingleLayer`: Sparse block of the normal derivative of the single layer integral operator
- `SpBlockDnDoubleLayer`: Sparse block of the normal derivative of the double layer integral operator
- `SpBlockSingleLayer`: Sparse block of the single layer integral operator
- `SpBlockIdentity`: Sparse block of the identity operator

IntOperators/Sparse/Interface/Full

- `SpSingleLayer`: Sparse matrix of the single layer integral operator
- `SpDnDoubleLayer`: Sparse matrix of the normal derivative of the double layer integral operator
- `SpDnSingleLayer`: Sparse matrix of the normal derivative of the single layer integral operator
- `SpDoubleLayer`: Sparse matrix of the double layer integral operator
- `SpIdentity`: Sparse matrix of the identity operator
- `SpPrecondDirichlet`: Sparse matrix of the preconditioned integral operator (sound soft case)
- `SpPrecondNeumann`: Sparse matrix of the preconditioned integral operator (sound hard case)

PostProcessing/FarField

- `FarField_to_RCS`: Radar Cross Section (RCS) from far field
- `FarField`: Generic far field computation from densities
- `RCS`: Generic Radar Cross Section (RCS) computation from densities

PostProcessing/FarField/Interface

- `RCSDoubleLayer`: Radar Cross Section (RCS) of the double layer potential only
- `RCSSingleLayer`: Radar Cross Section (RCS) of the single layer potential only
- `FarFieldSingleLayer`: Far field of the single layer potential only
- `FarFieldDoubleLayer`: Far field of the double layer potential only

PostProcessing/Geometry

- `MaskMatrixObstacles`: Matrix with boolean values inside or outside obstacles
- `BoundaryOfObstacles`: Extract the boundary of the obstacle
- `PlotCircles`: Display obstacles on figure

PostProcessing/IncidentWave

- `IncidentWaveOnGrid`: Compute incident wave on a (Matlab) meshgrid

PostProcessing/NearField

- `InternalPotential`: Compute potentials (single, double or linear combination) on a (Matlab) meshgrid and inside the obstacles
- `ExternalPotential`: Compute potentials (single, double or linear combination) on a (Matlab) meshgrid and outside the obstacles

PostProcessing/NearField/Functions

- `BlockPotential`: Generic block potential matrix used to compute external or internal potentials
- `GetPotentialOptions`: Options for potential computations are condensed here

PostProcessing/NearField/Interface

- `InternalSingleLayerPotential`: Internal potential of single layer potential only
- `ExternalSingleLayerPotential`: External potential of single layer potential only
- `ExternalDoubleLayerPotential`: External potential of double layer potential only
- `InternalDoubleLayerPotential`: Internal potential of double layer potential only

PreProcessing/Fourier

- `FourierTruncation`: Provide the number of mode to kept in the Fourier series

PreProcessing/Geometry

- `TriangularLattice`: Build a triangular lattice of disks
- `CheckPlacement`: Verify if the obstacles are satisfying the condition (overlapping, ...)
- `RectangularLattice`: Build a rectangular lattice of disks
- `RemoveDisk`: Remove some disks
- `CreateRandomDisks`: Place randomly random disks in a box

PreProcessing/IncidentWave

- `IncidentWave`: Full vector of a generic incident wave (right hand side)
- `BlockIncidentWave`: Block vector of a generic incident wave (right hand side, block=1 obstacle)

PreProcessing/IncidentWave/Block

- `BlockPointSource`: Block vector (=1 obstacle) of a the right hand side of the trace of a point source wave
- `BlockPlaneWavePrecond`: Block vector (=1 obstacle) of a the right hand side of the trace of a plane wave for the preconditioned problem of sound-soft scattering

- `BlockDnPlaneWave`: Block vector (=1 obstacle) of a the right hand side of the normal derivative of a plane wave
- `BlockPlaneWave`: Block vector (=1 obstacle) of a the right hand side of the trace of a plane wave
- `BlockDnPointSource`: Block vector (=1 obstacle) of a the right hand side of the normal derivative of a point source wave
- `BlockDnPlaneWavePrecond`: Block vector (=1 obstacle) of a the right hand side of the normal derivative of a plane wave for the preconditioned problem of sound-hard scattering

PreProcessing/IncidentWave/Full

- `PlaneWavePrecond`: Full vector (=all obstacle) of a the right hand side of the trace of a plane wave for the preconditioned problem of sound-soft scattering
- `PlaneWave`: Full vector (=all obstacle) of a the right hand side of the trace of a plane wave
- `DnPlaneWave`: Full vector (=all obstacle) of a the right hand side of the normal derivative of a plane wave
- `DnPointSource`: Full vector (=all obstacle) of a the right hand side of the normal derivative of a point source wave
- `PointSource`: Full vector (=all obstacle) of a the right hand side of the trace of a point source wave
- `DnPlaneWavePrecond`: Full vector (=all obstacle) of a the right hand side of the normal derivative of a plane wave for the preconditioned problem of sound-hard scattering

Bibliography

[1] F. Alouges, S. Borel, and D.P. Levadoux. A stable well-conditioned integral equation for electromagnetism scattering. *Journal of Computational and Applied Mathematics*, 204(2):440 – 451, 2007.

[2] X. Antoine, C. Chniti, and K. Ramdani. On the numerical approximation of high-frequency acoustic multiple scattering problems by circular cylinders. *J. Comput. Phys.*, 227(3):1754–1771, 2008.

[3] X. Antoine and M. Darbas. Alternative integral equations for the iterative solution of acoustic scattering problems. *Quaterly J. Mech. Appl. Math.*, 1(58):107–128, 2005.

[4] X. Antoine and M. Darbas. Generalized combined field integral equations for the iterative solution of the three-dimensional Helmholtz equation. *M2AN Math. Model. Numer. Anal.*, 1(41):147–167, 2007.

[5] X. Antoine and M. Darbas. *Integral Equations and Iterative Schemes for Acoustic Scattering Problems.* to appear, 2014.

[6] X. Antoine, K. Ramdani, and B. Thierry. Wide frequency band numerical approaches for multiple scattering problems by disks. *J. Algorithms Comput. Technol.*, 6(2):241–259, 2012.

[7] A. Bendali and M. Fares. Boundary integral equations methods in acoustics in computational acoustic scattering. In *Computational Methods for Acoustics Problems*, pages 1–36. Saxe-Coburg Publications, 2008.

[8] T. Betcke, S. N Chandler-Wilde, I. G. Graham, S. Langdon, and M. Lindner. Condition number estimates for combined potential boundary integral operators in acoustics and their boundary element discretisation. *Numerical Methods for Partial Differential Equations*, 27(1):31–69, 2011.

[9] S. Borel. *Résolution des équations intégrales pour la diffraction d'ondes acoustiques et électromagnétiques. Stabilisation d'algorithmes itératifs et aspects de l'analyse numérique.* PhD thesis, Université Paris XI, 2006.

[10] H. Brakhage and P. Werner. Über das Dirichletsche Aussenraumproblem für die Helmholtzsche Schwingungsgleichung. *Arch. Math.*, 16:325–329, 1965.

[11] A. J. Burton and G. F. Miller. The application of integral equation methods to the numerical solution of some exterior boundary-value problems. *Proc. Roy. Soc. London. Ser. A*, 323:201–210, 1971. A discussion on numerical analysis of partial differential equations (1970).

[12] S. N. Chandler-Wilde, I. G. Graham, S. Langdon, and M. Lindner. Condition number estimates for combined potential boundary integral operators in acoustic scattering. *J. Integral Equations Appl.*, 21(2):229–279, 2009.

[13] S. N Chandler-Wilde, I. G. Graham, S. Langdon, and E. A. Spence. Numerical-asymptotic boundary integral methods in high-frequency acoustic scattering. *Acta Numerica*, 21(1):89–305, 2012.

[14] D. L. Colton and R. Kress. *Integral Equation Methods in Scattering Theory.* Pure and Applied Mathematics (New York). John Wiley & Sons Inc., New York, 1983. A Wiley-Interscience Publication.

[15] M. Darbas. *Préconditionneurs Analytiques de type Calderòn pour les Formulations Intégrales des Problèmes de Diffraction d'Ondes.* PhD thesis, INSA de Toulouse, 2004.

[16] R. Harrington and J. Mautz. H-field, E-field and combined field solution for conducting bodies of revolution. *Archiv Elektronik und Uebertragungstechnik*, 4(32):157–164, 1978.

[17] R. Kress and W. T. Spassov. On the condition number of boundary integral operators for the exterior Dirichlet problem for the Helmholtz equation. *Numer. Math.*, 42(1):77–95, 1983.

[18] P. A. Martin. *Multiple Scattering. Interaction of Time-Harmonic Waves with N Obstacles*, volume 107 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 2006.

[19] J. Mautz and R. Harrington. A combined-source solution for radiation and scattering from a perfectly conducting body. *Antennas and Propagation, IEEE Transactions on*, 27(4):445 – 454, jul 1979.

[20] W.C.H. McLean. *Strongly elliptic systems and boundary integral equations.* Cambridge University Press, 2000.

[21] J.-C. Nédélec. *Acoustic and Electromagnetic Equations. Integral Representations for Harmonic Problems*, volume 144 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 2001.

[22] Y. Saad and M. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[23] B. Thierry. *Analyse et Simulations Numériques du Retournement Temporel et de la Diffraction Multiple.* Nancy University, Thèse de Doctorat, 2011.

[24] B. Thierry. A remark on the single scattering preconditioner applied to boundary integral equations. *Journal of Mathematical Analysis and Applications*, 413(1):212 – 228, 2014.